POPULAR EVENT ANALYSIS IN SOCIAL COMMUNITIES

BY

XIDE LIN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Professor Jiawei Han, Chair
Professor Chengxiang Zhai
Professor Bruce Schatz
Professor Qiaozhu Mei, University of Michigan

Abstract

The prevailing of Web 2.0 techniques has led to the boom of various online communities. Good examples are social communities such as Twitter, Facebook, Google+, and LinkedIn, which successfully facilitate the information creation, sharing, diffusion, and evolution among web users. As a result, a popular topic or event can spread much faster than in the Web 1.0 age. Indeed, when searching for a recent popular event (e.g., Hurricane Irene or Toyota Recall) on Twitter, all the results returned on the first page are created within the past five minutes.

In such a scenario, the objective of my thesis is to advance the data mining technique to create a system that detects, tracks, and analyzes the evolution and diffusion of popular events in a social community. Specially, in the first part of the dissertation, I introduce a mining algorithm for popular event detection, which can efficiently and effectively extract widely adopted and meaningful patterns of user behaviors; in the second part, I depict a novel and principled probabilistic model to track the popularity index of events in a time-variant social community that consists of both dynamic textual and structural information; in the third part of the dissertation, I address the problem of topic diffusions by studying the joint inference of topic diffusion and evolution in social communities, where contents and linkages in user-generated text information, together with social network structures, are used to facilitate the identification of topic adoption, the tracking of topic evolution, and the estimation of actual diffusion paths of any arbitrary topic.

To my family and my friends.

Acknowledgments

I would like to thank all the people who have helped and supported me during my Ph.D. study in the past several years.

First and foremost, I would like to express my deepest gratitude to my advisor Professor Jiawei Han for his help throughout my Ph.D. study. His vision, patience, enthusiasm and encouragement inspire me to think deeply and do solid work. This thesis would not have been possible without his support.

Also I would like to thank other doctoral committee members, Professor Qiaozhu Mei, Professor Chengxiang Zhai, and Professor Bruce Schatz, for their invaluable help on my research and constructive suggestions on the dissertation.

During my Ph.D. study, it is my great honor to work with talented researchers. I owe sincere gratitude to my colleagues in Database and Information System (DAIS) group, especially, Chen Chen, Bolin Ding, Yunliang Jiang, and Tim Weninger (sorted alphabetically), and researchers from NASA, especially, Nikunj C. Oza and Asok Srivastava (sorted alphabetically). I also thank all my other collaborators, Rick Barber, Liangliang Cao, Mihai Christodorescu, Marina Danilevsky, Hongbo Deng, Matt Fredrikson, Fabio Fumarola, Xin Jin, Zhenhui Li, Binbin Liao, Bing Liu, Jiebo Luo, Donato Malerba, Yizhou Sun, Tianyi Wu, Xifeng Yan, Yintao Yu, Duo Zhang, and Feida Zhu (sorted alphabetically).

Last but not least, I am indebted to my mother and my family for their care and support all the time.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The prevailing of Web 2.0 techniques has led to the boom of various online communities. Good examples are social communities such as *Twitter*[1], *Flickr*[2], *Facebook*[3] and *LinkedIn*[4], which successfully facilitate the information creation, sharing, diffusion, and evolution among web users. As a result, a popular topic or event can spread much faster than in the Web 1.0 age. Indeed, when searching for a recent popular event (*e.g.*, 'Hurricane Irene' or 'Google Plus') on Twitter, all the results returned on the first page are created within the past five minutes.

In many scenarios, it is appealing to have a system that detects, tracks, and analyzes the evolution and diffusion of popular events in a social community. What are the hot topics discussed on Twitter today? Who initialized a rumor? Who are still interested in watching *Avatar* 50 days after its release date? What do people say about *Tiger Woods* before and after the scandal? Hot topics emerge, prevail, and die, following the connected social network of users. It is desirable to monitor what people like, whether they like, and how the interest change over time. It is also interesting to analyze such online communities with understanding the cascading behaviors and the diffusion of topics.

Tracking the evolution of a popular topic is challenging. The diffusion of a behavior is clear, *e.g.*, it is quite easy to identify whether the behavior of 'having McDonalds for lunch' happened today. But the diffusion of an event is vague: you do not know whether I am interest in an event, *e.g.*, how could you know that I am actually interested in 'Tiger Woods' if I mentioned 'Mistress

[1] http://www.twitter.com
[2] http://www.flickr.com
[3] http://www.facebook.com
[4] http://www.linkedin.com

counts up to 9'? Even you do, from whom did I get this interest?

Fortunately, a large volume of text data is generated from the social communities. Besides communicating with friends, a web user also constantly generates text contents such as blogs, tweets, feeds and comments. Both the communications and the contents are changing along time, resulting in a network structure and a text collection which evolve simultaneously and interrelatedly. When we read what you have written, we can infer your interest in an event; and when we glimpse your communications, we can guess where the interest comes from. When we track the communications and contents over time, we can find out the burstiness, the evolution, and the spread of an event in a social community. Taking another example, researchers regularly publish papers and also collaborate with other researchers. By simultaneously analyzing the evolution of publications and co-authorship, we can track how a research topic initializes, changes, and diffuses over the research community, in terms of both paper contents and its impact. In all these scenarios, there is an urgent need for a principled method that couples a stream of text and a stream of networks in order to track popular events.

Another important and even more challenging problem is concerned with understanding the cascading behaviors and the diffusion of information. Epidemic diseases, adoption of innovation, memes of information, and many types of user actions all spread widely in social communities, following the social network of users. The modeling of information diffusion plays a crucial role in many domains: The contagion of disease forms the foundation of epidemics; the social influence in cascading behaviors has been a basic mechanism of viral marketing; and the diffusion of topics is essential to the understanding of scientific innovation.

Many studies are done in scenarios where the actual contagion/diffusion paths are observed. Such an assumption, which is considered as a common practice in user surveys and controlled user studies, does not apply to large scale online communities however. While the adoptions of behaviors are relatively easy to observe, the evidence of actual contagion and influence tend to be vague. Who infected whom? Who got the gossip from whom? Who influenced whose research? There are still substantial challenges in this micro-level analysis of information diffusion in large

scale social networks. Indeed, users who joined a community or purchased an iPad usually won't explain which particular friends have influenced them; rumor spreaders tend to cover the source of the information; a researcher cites many references in her paper, without labeling the top three that have the most salient influence on her work. The identification of contagion is difficult even if the general social network structure is observed. It is a non-trivial task to detect the actual diffusion paths of user behaviors merely based on the time of adoption and the social network structure, known as the problem of diffusion (or influence) inference [22].

Organization. In the first part of the dissertation, I introduce a mining algorithm for popular event detection, which can efficiently and effectively extract widely adopted and meaningful patterns of user behaviors

In the second part, I depict a novel and principled probabilistic model (called *PET*) for *Popular Events Tracking* in a time-variant social community that consists of dynamic textual and structural information. Specifically, *PET* takes (i) a stream of document collections, (ii) a stream of network structures, and (iii) a popular event, as the input, and generates analysis on the event in both aspects of popularity and content evolution.

In the third part, I address the problem of topic diffusions by studying the joint inference of topic diffusion and evolution in social communities. Content and linkage in user-generated text information, together with social network structures, are used to facilitate the identification of topic adoption, the tracking of topic evolution, and the estimation of actual diffusion paths of any arbitrary topic.

The conclusions and summaries are given in the last chapter.

Chapter 2

Related Work

As my thesis focuses on popular event analysis in social communities, it is related to the study of *event detection*, *event tracking*, and *information diffusion*, which also involves work in *topic modeling* and *pattern mining*. A brief overview of these related methods is discussed in this section

2.1 Event Detection

There have been extensive studies on event detection, which facilitate a wide range of tasks such as search [46], clustering [65], classification [24], stock prediction [91], and event tracking [56]. Rattenbury, *et al.* [69] cast photo tags on Flickr into a multi-dimensional space according to their semantic, temporal and location attributes, where a dense area is detected as an event. Similarly, Chen, *et al.* [15, 16] extract events from the click-through data by considering the semantic dimension and temporal dimension of queries. Zhao, *et al.* [96] model temporal data by a query-page bipartite graph, and an event (represented by a query-page pair) is detected by clustering on graphs. Li, *et al.* [53] detect events from news articles with a generative model of content, time, locations and people, while Zhao, *et al.* [97] combine text-based clustering, temporal segmentation, and information flow-based graph cuts. [46, 101, 42, 67, 35] targets on identification of irregular behaviors in data streams, either by applying thresholds [46, 42, 35], or reporting windows with abnormal aggregates [101] or significance [67].

Another line related to event detection is mining correlated patterns. Kim, *et al.* [39] propose to mine closed correlated patterns in order to reduce candidate patterns produced without information loss. Zhou, *et al.* [99] combine association with correlation in the mining process to discover

both association and correlation rules. By adopting the FP-tree data structure, He, *et al.* [31] mine the top-k strongly correlated item pairs without a minimum correlation threshold, where k is the desired number of pairs that have the largest correlation values. Ke, *et al.* [37] extract correlations from quantitative databases efficiently by utilizing normalized mutual information and all-confidence to perform a two-level pruning. They show that mining correlations is more effective than mining associations. The advantages of correlated patterns over associations are also discussed by Jiang, *et al.* [34]. Younes, *et al.* [94] introduce a new concise representation of frequent correlated patterns associated with the *bond* measure. The proposed representation allows not only to efficiently derive the correlation rate of a given pattern, but also to exactly offer its conjunctive, disjunctive and negative supports. Additional, in the graph pattern mining literature, there is also some work on mining correlated and representative graph patterns by Chen, *et al.* [14] and Ke, *et al.* [38, 36].

2.2 Topic Modeling

Topic modeling approaches introduced by Hofmann, *et al.* [32] and Blei, *et al.* [9] have been developed to mine variations of topics in different contexts [78, 52, 21, 61, 25], evolution of topics [61, 8, 87, 74], and correlated patterns in multiple text streams [88]. These methods generally do not consider the network structures, and thus could not been applied to analyze popular events in social communities.

Recently, incorporating network regularization in topic modeling has been proposed [12, 60, 79, 56, 55]. Cai, *et al.* [12] utilize Laplacian to develop a model called PLSI, while Mei, *et al.* [60] use a harmonic function to enforce the constraint that topic distribution on neighboring nodes should be similar. Sun, *et al.* [79] define a Markov Random Field on the graph to model the influence between nodes in a generative way. Lin, *et al.* [56] leverage Gibbs Random Field to estimate the popularity index of a textual topic depending on social connections and history. Tang, *et al.* [83] model the influence graph among users by considering both network structures and

topics. Weng, *et al.* [89] find experts at Twitter for specific topics.

2.3 Event Tracking

A state automation model was proposed by Kleinberg, *et al.* [43] to detect bursty activities from an email arrival stream, by assuming the rates of messages are determined by underlying hidden states. Ihler, *et al.* [33] model the sequence of counting data by combining two Poisson distributions - one for the normal periodic count data and the other for the rare events. Araujo, *et al.* [5] propose a model for detecting and tracking events in a discrete temporal sequence, which transforms Bayesian inference into the Potts model in statistical physics, and finds the configuration that minimizes the Potts energy. He, *et al.* [30] analyze word trajectories in both time and frequency domains, with the specific goal of identifying important and less-reported, periodic and aperiodic words. Morris, *et al.* [62] evaluate network diffusion models by considering the question that when a local behavior can spread to the whole population. These methods typically take either sequences of statistical data (*e.g.*, word frequencies) or interaction systems as the input, but do not simultaneously consider network structures and textual topics in the data stream generated by social communities, which are shown in later section of this proposal as quite effective in tracking popular events in social communities.

2.4 Information Diffusion

Information diffusion [7, 27, 41, 85, 54, 22, 49, 50, 62, 49, 6, 27] is a classic topic in social network analysis, which models the cascade of behaviors on a network structure. Kimura, *et al.* [40] aim at blocking a small subset of links so as to minimize the spread of contamination, while Chen, *et al.* [18, 17] find a small subset of nodes that maximize the spread of behaviors under certain diffusion model. Tang, *et al.* [83] and Liu, *et al.* [57] estimate social graphs with edges labeled with probabilities of influence between users. Gomez-Rodriguez, *et al.* [26] estimate the best

latent network structure, through which a set of events are spread well. Kimura, *et al.* [41] predicts the sharing scale of an opinion, while Tang, *et al.* [83], Goyal, *et al.* [1] and Lee, *et al.* [47] extract influential nodes or sub-graphs. Zhou, *et al.* [98] model the social interactions and topic evolutions in an academic network.

This line of work, however, do not consider textual topics, and usually do not consider the evolution of ties. It is thus hard to be applied to tracking and analyzing popular events.

Chapter 3

Popular Event Detection

In this section, I propose a mining algorithm for popular event detection in social communities. Specifically, I study the task of finding sequential patterns (*e.g.*, a topic represented by a sequence of keywords, or a tourism path in San Francisco) from social user generated information, which are not only popular but also meaningful events.

As stated in Section 1, mining correlated patterns is different from association patterns, which is more complicated and challenging. The reason is, user-generated information in real social networks is usually huge. As a result, when the *minimum support* threshold is high, only obvious common sense 'knowledge' will be found; when *minimum support* is set low, a huge number of patterns will be generated, a majority of which are redundant, uninformative or just random combinations of popular data objects [48]. Let us start with a toy example to show (i) what are *the differences between an association and a correlated pattern*, and (ii) how a *reasonable correlation measure can effectively mine correlated sequential patterns*.

Example 3.0.1. *Suppose we have a mini sequence database made up of 16 word phrases extracted from bibliographical records (see Table 3.1). Some phrases therein are research topics, e.g., 'support vector machine' and 'machine learning', while others are combinations of popular terms, e.g., 'support machine', 'graph mining' and 'clustering algorithm'. All of the five patterns are association patterns because they appear together frequently, but only the first two are correlated patterns, as they express specific meaningful research topics, whereas the other three express either useless or overly broad meanings.*

Although the answer to whether a sequential pattern is correlated or not is not an absolute 'Yes' or 'No', we at least expect to match common knowledge, i.e., the phrase 'support vector

S_1	support vector machine
S_2	graph support classification
S_3	support vector machine
S_4	graph theory
S_5	support evidence
S_6	graph pattern mining
S_7	machine learning
S_8	graph pattern mining
S_9	spectral clustering algorithm
S_{10}	sequence pattern mining
S_{11}	spectral clustering algorithm
S_{12}	novel association pattern mining
S_{13}	spectral clustering method
S_{14}	construction algorithm
S_{15}	spectral clustering model
S_{16}	EM algorithm

Table 3.1: The Example Sequence Database $\mathcal{SDB}$

machine' would be more correlated than 'support vector'. Thus, under an appropriate measure of correlation, a long pattern should be allowed to be more correlated than its sub-patterns. Based on such observation, we will re-examine a lot of interestingness measures and make careful selections for our mining task in latter sections.

Organization The rest of this section is organized as follows. Section 3.1 formally defines the problem of mining frequent correlated sequential patterns in social communities, and analyzes the usage of correlation measures. As a solution, a novel mining method based on pattern growth methodology is proposed in Section 3.2. Finally, we present experiments and results in Section 3.3.

3.1 Preliminaries

In this section, we formally define the problem of mining correlated sequential patterns (Section 3.1.1), and theoretically and empirically analyze certain properties under this definition, in order to select appropriate association measure(s) for our mining task (Section 3.1.2).

3.1.1 Problem Formulation

Let $\mathcal{E}$ be a set of distinct *items*. A sequence S is an ordered list of items, denoted as $S = e_1\ e_2\ \cdots\ e_{|S|}$, where $e_i \in \mathcal{E}$ is an item. For convenience, we refer to the i^{th} item e_i in the sequence S as $S[i]$. An input *sequence database* is a set of sequences, denoted as $\mathcal{SDB} = \{S_1, S_2, \cdots, S_N\}$, where each sequence S_i is called a *transaction* in the database $\mathcal{SDB}$.

Definition 3.1.1. *(Subsequence)* For two sequences $S = e_1\ e_2\ \cdots\ e_{|S|}$ and $S' = e'_1,\ e'_2,\ \cdots,\ e'_{|S'|}$ ($|S| \leq |S'|$), S is said to be a *subsequence* of S', denoted by $S \subseteq S'$, if there exists a series of one-to-one mapping positions $1 \leq p_1 < p_2 < \cdots < p_{|S|} \leq |S'|$, s.t., $S[i] = S'[p_i]$ (*i.e.*, $e_i = e'_{p_i}$) for any $i \in 1, 2, \cdots, |S|$. In particular, for $|S| < |S'|$, we call S a *proper (strict) subsequence* of S', denoted by $S \subset S'$. We may also say S' is a *super-sequence* of S, or S' contains S.

A *pattern* P is also a sequence. For two patterns P and P', if P is a subsequence of P', then P is said to be a sub-pattern of P', and P' is a super-pattern of P.

Definition 3.1.2. *(Projected Database, Support, and Probability)* For an input sequence database $\mathcal{SDB}$ and a pattern P, $\mathcal{DB}(P)$) is the set of transactions in $\mathcal{SDB}$, of which P appears as a subsequence. We define the *support* of P as $Sup(P) = |\mathcal{DB}(S)|$ and therefore the *probability* of P as $Pr(P) = \frac{Sup(P)}{|\mathcal{SDB}|}$. Any transaction in $\mathcal{DB}(P)$ is referred as P's *supporting transaction* and $\mathcal{DB}(P)$ is called the *projected database* of $\mathcal{SDB}$ based on the pattern P.

Definition 3.1.3. *(Cutting and Cutting Set)* For a sequence S ($|S| > 2$) and an ordered list of subsequences $C = \{c_1,\ c_2,\ \cdots,\ c_{|C|}\}$ ($|C| \geq 2$), we call C a cutting of S, if any $c_i \in C$ is non-empty (*i.e.* $|c_i| > 0$) and the concatenation of $c_1, c_2, \cdots, c_{|C|}$ equals to S. Specifically, C is k-piece cutting if $|C| = k$. We abbreviate the set of all k-piece cuttings of S as $\mathcal{C}_k(S)$, and furthermore $\mathcal{C}_{2..k}(S) = \mathcal{C}_2(S) \cup \cdots \cup \mathcal{C}_k(S)$.

A number of interesting correlation measures defined on association patterns [4, 10] have been proposed and analyzed [92, 80], including χ^2, *lift*, *all-confidence*, *max-confidence*, *Kulcsynski* and *Cosine*. We use the notation $Cor(P)$ to denote the correlation score (according to

any correlation measure) of a pattern P. Formally, a sequential pattern P is said to be *frequent* if $Sup(P) \geq min\_sup$ and *correlated* if $Cor(P) \geq min\_cor$, where min_sup and min_cor are specified empirically by users. *The mining task of frequent correlated sequential patterns is to find the complete set of sequential patterns which are both frequent and correlated.*

3.1.2 Measure Selection in the Scenario of Social Communities

In this chapter, we discuss the relationship between frequent patterns and popular events in the scenario of social network, based on which we make selections on appropriate correlation measures for our mining task.

Measure	Definition	Average Connectivity Ratio
f	$sup(p)$	8.89%
lift	$\frac{Pr(p)}{Pr(a_1)Pr(a_2)\cdots Pr(a_n)}$	**11.03**$\%$
all-confidence	$\frac{sup(p)}{max\{sup(a_1),sup(a_2),\cdots,sup(a_n)\}}$	9.76%
max-confidence	$\max\{\frac{sup(p)}{sup(a_1)}, \frac{sup(p)}{sup(a_2}, \cdots, \frac{sup(p)}{sup(a_n)}\}$	9.50%
Cosine	$\frac{sup(p)}{\sqrt{sup(a_1)sup(a_2)\cdots sup(a_n)}}$	9.99%
Kulc	$\frac{sup(p)}{n}\left(\frac{1}{sup(a_1)} + \frac{1}{sup(a_2)} + \cdots + \frac{1}{sup(a_n)}\right)$	10.44%

Table 3.2: The experimental results on average connectivity ratio

As mentioned in Section 1, a time-variant social communities is consisted of both a stream of text information and a stream of dynamic structures. Hence, if a sequence of social behaviors is regarded as a popular event, it should not only have been performed by social users for sufficient times, but also have strong tie among its related social users. According to this heuristic, (i) we have a review on six measures that are popularly used in pattern mining problems (see the first two columns of Table 3.2); (ii) frequent association patterns are extracted by the PrefixSpan [66] algorithm from paper titles in the DBLP dataset (see Section 3.3); (iii) for each frequent pattern $p = a_1 a_2 \cdots a_n$, by considering its supporting transactions d (*i.e.*, the papers that contain p in their titles) and its related authors a (*i.e.*, the authors of the papers in d), we calculate the ratio of the connectivities among a in d against the connectivities among a in all their publications, called

the connectivity ratio of p; (iv) frequent patterns are ranked according to each measure, and the average connectivity ratio of the top five patterns is listed in the third column of Table 3.2.

Based on the above experimental results, the *lift* measure defined on association patterns (Equation 3.1) achieves the highest average connectivity ratio, for which we regard it as the most appropriate correlation measure in our mining tasks. Moreover, we extend it to the correlation measure on sequential patterns as Equation 3.2.

$$lift(a_1 a_2 \cdots a_n) = \frac{Pr(a_1 a_2 \cdots a_n)}{\prod_{i=1..n} Pr(a_i)} \tag{3.1}$$

$$\begin{aligned} Cor(S) &= \frac{Pr(S)}{\underset{C=\{c_1,\cdots,c_{|C|}\}\in\mathcal{C}_{2..|S|}(S)}{Max}\{\prod_{i=1..|C|} Pr(c_i)\}} \\ &= \underset{C=\{c_1,\cdots,c_{|C|}\}\in\mathcal{C}_{2..|S|}(S)}{Min}\{\frac{Pr(S)}{\prod_{i=1..|C|} Pr(c_i)}\} \end{aligned} \tag{3.2}$$

The general ideas of the traditional measure *lift* defined on itemsets (Equation (3.1)) and the measure *cor* we proposed for sequential patterns (Equation (3.2)) are the same: the probability of a correlated pattern should be significantly larger than the joint probability of 'information units' that make up the correlated pattern. However, their difference relies on: the only possible information units considered by *lift* are single items appearing in the pattern, while the concept of information unit in *cor* is extended to any sub-pattern of the correlated pattern, so that the *cor* measure considers more completely than the *lift* measure.

We expressed the information unit by 'cutting' (Definition 3.1.3). A correlated sequential pattern P is expected to fail the null hypothesis consisting of any arbitrary cutting that make up P, *i.e.*, the probability of a correlated pattern P should be significantly larger than the product of the probabilities of sub-patterns appearing in any cutting of P. Hence, we use the ratio between the probability of P and the maximum of the joint probability of its sub-patterns appearing separately in any possible cutting of P to denote the correlation score of P.

3.2 Efficient Mining Algorithm

In this section, we first introduces the classical *PrefixSpan* [66] algorithm (Section 3.2.1), based on which a three-stage mining method is developed to extract correlated sequential patterns with its complexity discussed in Section 3.2.2.

3.2.1 The PrefixSpan Algorithm

Let us begin with introducing two concepts *prefix* and *suffix* as below, which are very important to the mining algorithm.

Definition 3.2.1. *(Prefix and Suffix)* For two sequences $S = e_1\ e_2 \cdots\ e_n$ and $S' = e'_1\ e'_2 \cdots\ e_m$ ($n \leq m$), S is said to be a n-sized *prefix* of S' if $e_i = e'_i$ for every $i \in 1, 2, \cdots, n$. In particular, for the case that $n = m - 1$, S is the *immediate prefix* of S'. Similarly, S is said to be a n-sized *suffix* (*postfix*) of S' if $e_i = e'_{m-n+i}$ for every $i \in 1, 2, \cdots, n$, and S is the *immediate suffix* of S' if $n = m - 1$.

PrefixSpan [66] belongs to the series of pattern-growth methodology [29, 93]. The major idea is that, instead of projecting sequence databases by considering all the possible occurrences of frequent subsequences, the projection is based only on frequent prefixes because any frequent subsequence can always be found by growing a frequent prefix. Generally speaking, sequential patterns can be mined by a prefix-projection method in three steps:

1. Find all frequent length-1 patterns,
2. Divide search space into projected databases based on prefixes, and
3. Mine each projected database recursively and output sequential patterns with prefixes added to the front.

Due to space limitation, please refer to [66] for more technical details.

However, *there are challenges to compute the correlation score (Equation (3.2)) of a frequent pattern during the mining procedure of PrefixSpan, because we do not know the probability of any of its sub-patterns*. The easy and straightforward solution could be: we generate all frequent patterns first, then create an in-memory index on these patterns, and re-examine frequent patterns by calculating their correlation scores. However, such a solution is undesirable for several reasons:

1. First, to avoid missing useful patterns, the minimum support is usually set low, as a result of which, the in-memory index on frequent patterns may exceed the availability of the primary memory. The situation happens in our experiments (Section 3.3).

2. Second, even if we are able to create an in-memory index for frequent patterns, the efficiency of the pattern mining algorithm will heavily rely on the efficiency of the hash function of the index (*i.e.*, the cost of accessing one value), especially if the patterns themselves may have various formats and structures.

3. Third, when an algorithm is disk-based, it is easier to accelerate by parallel computing [3, 86]. We will have a short discussion in Section 3.2.2 about how to accelerate our algorithm by utilizing distributive computation.

In the remained of this section, we propose an efficient mining algorithm, which could avoid creating an in-memory index for frequent patterns.

3.2.2 The PSBSpan Mining Algorithm

Let us start with several key concepts:

Definition 3.2.2. (k-Piece Correlated Pattern and k-Piece Maximum Cutting Probability) A frequent pattern P is called a k-piece correlated pattern ($k \geq 2$) if $k\text{-}Cor(P) \geq min\_cor$, where

k-$Cor(P)$ is derived from Equation (3.2) by setting a constraint on the cutting size, *i.e.*,

$$k\text{-}Cor(P) = \frac{Pr(P)}{\underset{C=\{c_1,c_2,\cdots,c_{|C|}\}\in\mathcal{C}_{2..k}(P)}{Max}\{\prod_{i=1}^{|C|} Pr(c_i)\}} \tag{3.3}$$

For convenience, we call the denominator of the first line of Equation (3.3) the *k-piece maximum cutting probability*, defined as

$$k\text{-}Mcp(P) = \underset{C=\{c_1,c_2,\cdots,c_{|C|}\}\in\mathcal{C}_{2..k}(S)}{Max}\{\prod_{i=1..|C|} Pr(c_i)\} \tag{3.4}$$

In particular, we have k-$Mcp(P) = Pr(e)$ for any k, if $P = e$ is a single-item pattern.

Definition 3.2.3. (Prefix and Suffix Upper-bound) For a sequence $P = e_1\ e_2 \cdots\ e_n$ ($n \geq 2$), we estimate two upper-bounds of $Cor(P)$, called the *prefix upper-bound* (Equation (3.5)) and the *suffix upper-bound* (Equation (3.6)):

$$\begin{aligned} Cor_{pre}(P) &= \frac{Pr(P)}{Pr(e_1e_2\cdots e_{n-1})Pr(e_n)} \qquad (3.5)\\ &= \frac{Sup(e_n|\mathcal{DB}(e_1e_2e_{n-1}))}{|\mathcal{DB}(e_1,e_2,\cdots,e_{n-1})|Pr(e_n)} \\ Cor_{post}(P) &= \frac{Pr(P)}{Pr(e_2,e_3,\cdots,e_n)Pr(e_1)} \qquad (3.6)\\ &= \frac{Sup(e_1|\mathcal{DB}(e_2,e_3,\cdots,e_n))}{|\mathcal{DB}(e_2,e_3,\cdots,e_n)|Pr(e_1)}, \end{aligned}$$

where $Sup(e_i|\mathcal{DB}(P))$ is the count of item e_i in the projected database $\mathcal{DB}(P)$.

In this section, we will first introduce a three-stage mining method for 2-piece correlated patterns (Section 3.2.2), then extend the results to correlated patterns of pieces of any arbitrary size (Section 3.2.2), and finally analyze the overall time complexity (Section 3.2.2).

Algorithm 1 PrefixSpan($\mathcal{DB}(S)$, S, min_sup, min_cor)

Input:
the projected database $\mathcal{DB}(S)$,
the prefix string S,
the minimum support threshold min_sup, and
the minimum correlation threshold min_cor.

$CP = \emptyset$.
Count the support $Sup(e|\mathcal{DB}(S))$ of each item e in the database $\mathcal{DB}(S)$.
for each item e in $\mathcal{DB}(S)$ (enumerated according to lexicological order) **do**
  if $Sup(e|\mathcal{DB}(S)) \geq min\_sup$ **then**
    $S' = S + e$
    R = PrefixSpan($\mathcal{DB}(S')$, S', min_sup, min_cor)
    if $|S'| \geq min\_sup$ **then**
      if $\frac{Sup(e|\mathcal{DB}(S))}{|\mathcal{DB}(S)|Pr(e)} \geq min\_cor$ **then**
        output S' as a temporarily correlated pattern with related information
      end if
      if $R \neq \emptyset$ **then**
        output S' as a frequent pattern with related information
      end if
    end if
  end if
  return CP
end for

Mining 2-Piece Correlated Patterns

We generate the complete set of 2-piece correlated patterns in three steps: *PrefixSpan*, *SuffixSpan*, and *Binding*, called the *PSBSpan* algorithm.

PrefixSpan. This step is almost the same as the traditional *PrefixSpan* algorithm described in Section 3.2.1, but there is only one special thing we are doing here:

1. we calculate the prefix upper-bound of each frequent pattern, and a pattern is a said to be a **temporarily correlated patterns**, if its prefix upper-bound (Equation (3.5)) is no less than the minimum correlation threshold min_cor. We output a frequent pattern, only if (i) it is a temporarily correlated pattern, or (ii) it is a prefix of a temporarily correlated pattern. The reasons are:

   (a) If a pattern is not a temporarily correlated pattern, it is definitely not a correlated pattern, so

that it is unnecessary to consider it further. We will show in our experiments (Section 3.3) that empirically such pruning can filter out about half of frequent but non-correlated patterns.

(b) The cost of the pruning is cheap: when we are at the stage of a frequent pattern, we must have accessed the probability of its immediate prefix and trivial space is needed to store the probability of each prefix of the current pattern and of each single item.

2. For each pattern selected to output by the above step, we not only output its probability, but also the probability of any of its proper prefixes. We will show the utility of such an output in the *Binding* step.

The pseudo-code is depicted in Algorithm 1.

association	0.06	vector	0.13
machine	0.19	learning	0.06
spectral	0.25	clustering	0.25
algorithm	0.25	method	0.06
model	0.06	mining	0.25
classification	0.06	graph	0.25
EM	0.06	theory	0.06
pattern	0.25	novel	0.06
sequence	0.06	support	0.25
construction	0.06	evidence	0.06

Table 3.3: The Vocabulary $\mathcal{E}$ for Example 3.2.1

Example 3.2.1. *We demo the mining procedure of PrefixSpan with a toy sequence database consisting of the abbreviated title of* 16 *publications. The transactions and vocabulary (with word probabilities) are shown in Table 3.1 and Table 3.3, respectively. The parameters are empirically set to be* $min\_sup = 2$ *and* $min\_cor = 3.0$.

The frequent patterns are listed in the first column of Table 3.4 along with their supports. Prefix upper-bounds of those frequent patterns are calculated in the second column. It is shown that 10 *non-single-item patterns are generated during the mining procedure, only* 6 *(marked with ✓) of which have the prefix upper-bounds larger than the minimum correlation threshold* 3.0.

Non-Single-Item Patterns		*PrefixSpan*(Cor_{pre})		*SuffixSpan*(Cor_{post})		*Binding*
support vector	$sup = 2$	4.0	✓	4.0	✓	✓
support vector machine	$sup = 2$	5.3	✓	4.0	✓	✓
support machine	$sup = 2$	2.7		2.7		
vector machine	$sup = 2$	5.3	✓	5.3	✓	✓
spectral clustering	$sup = 4$	4.0	✓	4.0	✓	✓
spectral clustering algorithm	$sup = 2$	2.0		4.0	✓	
graph pattern	$sup = 2$	2.0		2.0		
graph pattern mining	$sup = 2$	4.0	✓	2.0		
graph mining	$sup = 2$	2.0		2.0		
pattern mining	$sup = 4$	4.0	✓	4.0	✓	✓

Table 3.4: The Sequential Patterns Extracted at Different Stages

Theorem 1. *The output sequences of Algorithm 1 are automatically sorted lexicologically.*

Proof. For two sequential patterns $S = e_1, e_2, \cdots, e_n$ and $S' = e'_1, e'_2, \cdots, e_m$ in the output, *w.l.o.g.*, we suppose $S < S'$ that $e_i = e'_i$ for any $i = 1, 2, \cdots, j$ and $e_{j+1} < e'_{j+1}$, *i.e.*, $S^* = e_1\ e_2\ \cdots\ e_j$ is the longest common prefix of S and S'. The two sequences are projected into two different sub-databases $\mathcal{DB}(S^* + e_{j+1})$ and $\mathcal{DB}(S^* + e'_{j+1})$, respectively, at the function call of *PrefixSpan*($\mathcal{DB}(S^*)$, S^*, min_sup, min_cor). Since *PrefixSpan* generates all patterns in $\mathcal{DB}(S^* + e_{j+1})$ before the patterns in $\mathcal{DB}(S^* + e'_{j+1})$, as a result, S is output earlier than S'. □

SuffixSpan. This step is a mirrored version of the *PrefixSpan* step, but with an additional sorting step at the end. Generally speaking, we

1. Project databases based on suffixes and generate patterns by concatenating suffixes to the end of patterns mined from projected databases.

2. Calculate the suffix upper-bound (Equation (3.6)) of each pattern, and output a frequent pattern if it is a temporarily correlated pattern or it is a suffix of a temporarily correlated pattern, together with the probability of its suffixes.

3. Sort the output sequences in lexicological order by (disk-based) sorting methods [44].

Example 3.2.2. *Following Example 3.2.1, the suffix upper-bounds (Equation (3.6)) are listed in the third column of Table 3.4 for each frequent pattern. 6 (marked with ✓) of the 10 frequent patterns*

have suffix upper-bounds larger than the minimum correlation threshold 3.0 *and therefore remain as temporarily correlated patterns.*

The pseudo-code is depicted in Algorithm 2.

Binding. With the two sorted output lists generated from the two previous steps, we

1. Find their overlapping set.
2. Do a final verification for each pattern in the overlapping set, to check whether it is a 'true' 2-piece correlated pattern according to Equation (3.3).
3. Output a frequent pattern if it is a 2-piece correlated pattern or it is a prefix of a 2-piece correlated pattern, along with the 2-piece maximum cutting probability of each of its prefixes.

The procedure is basically a merge sort (in linear time) [44], and the verification step is easy, since we have the probability of any prefix or suffix of a temporarily correlated pattern. For each 2-piece correlated pattern, we output its probability as well as the 2-piece maximum cutting probability of any of its proper prefixes.

Example 3.2.3. *After 'binding' the results generated from Example 3.2.1 and 3.2.2, the 'truly' correlated patterns are marked with ✓ in the last column of Table 3.4.*

Extending to k-Piece Correlated Patterns

To find correlated patterns of pieces of any arbitrary size, the only problem we need to solve relies on how to extend k-piece correlated patterns to $(k+1)$-piece ones. Let us start with two Theorems.

Theorem 2. *The* 2*-piece correlated patterns generated by PSBSpan are automatically sorted lexicologically.*

Proof. According to Algorithm 2, correlated patterns are generated at the *Binding* step only if they are contained in the result from the *PrefixSpan* step. Since results generated by the *PrefixSpan* step

are sorted lexicologically as per Thereom 1, and *Binding* does not change the order among patterns from the *PrefixSpan* step, the results from *PSBSpan* are also automatically sorted. □

Theorem 3. *For a k-piece correlated pattern P ($|P| \geq k$), if we have the k-piece maximum cutting probability (Equation (3.4)) of any of its proper prefixes and the probability of any of its proper suffixes, we can calculate the $(k+1)$-piece correlated score (Equation (3.3)) and the $(k+1)$-piece maximum cutting probability of P.*

Proof. We do the below formula transformation:

$$\begin{aligned}
& (k+1)\text{-}Mcp(P) \\
= & \underset{C=\{c_1,\cdots,c_{|C|}\}\in\mathcal{C}_{2..k+1}(P)}{Max}\{\prod_{i=1..|C|} Pr(c_i)\} \\
= & \underset{C,c':C+c'=P}{Max}\{\underset{C=\{c_1,\cdots,c_{|C|}\}\in\mathcal{C}_{2..k}(P)}{Max}\{\prod_{i=1..|C|} Pr(c_i)\}Pr(c')\} \\
= & \underset{C,c':C+c'=P}{Max}\{k\text{-}Mcp(C)Pr(c')\}
\end{aligned}$$

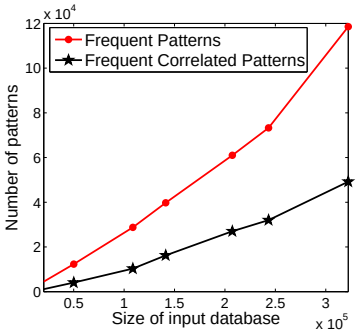
Given the (k+1)-piece maximum cutting probability of P (Equation (3.4)), it is easy to obtain its (k+1)-piece correlation score (Equation (3.3)), so as to judge whether P is a (k+1)-piece correlated pattern. □

As per Theorem 2, the 2-piece correlated patterns generated by Algorithm 2 are sorted, so that we could

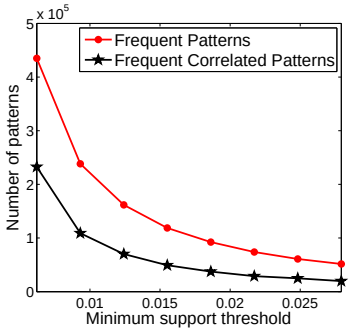
1. Find the overlapping set of patterns from Algorithm 2 and the patterns from the *Suffix* step (in linear time),

2. Calculate the 3-piece correlated score as per Theorem 3 for each pattern in the overlapping set, and judge whether it is a 'true' 3-piece correlated pattern. This procedure is also called a *Binding* step, since the pseudo-code can be derived from Algorithm 2 by just revising a few lines (*e.g.*, Line 16)[1].

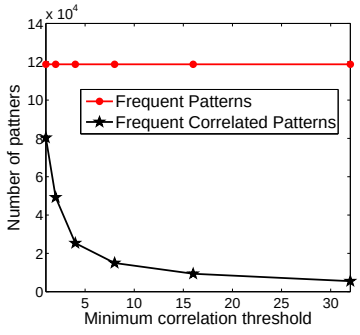
[1] We omit it here due to space limitation



(a) the number of patterns *w.r.t.* the size of database



(b) the number of patterns *w.r.t.* the minimum support threshold



(c) the number of patterns *w.r.t.* the minimum correlation threshold

Figure 3.1: Frequent Patterns V.S. Frequent Correlated Patterns

When k-piece correlated patterns are wanted, we just repeat the above procedure for a total of (k-1) iterations.

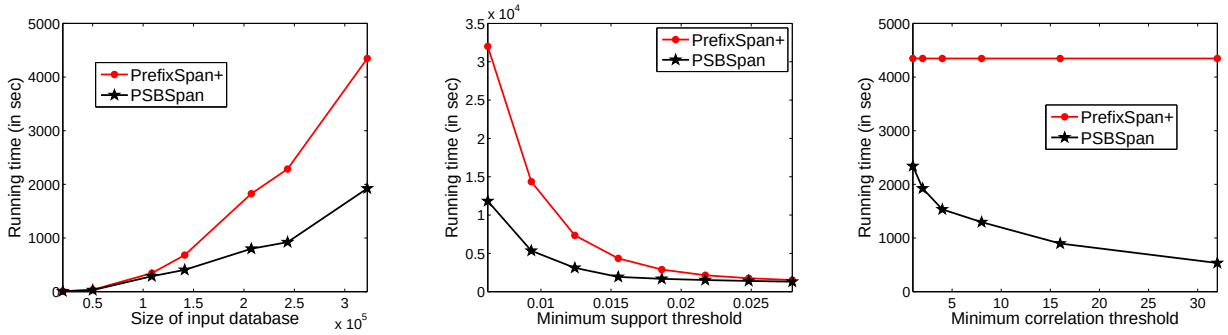
Complexity Analysis

Since *PrefixSpan* is a well-known algorithm [66], we simply assume the time complexity of the two steps are both $O(\delta)$, and the number of frequent patterns generated by the two steps is $O(\sigma)$.

The *SuffixSpan* step is equal to a mirrored version of the *PrefixSpan* step plus a disk-based sorting step [44] (the complexity is denoted as $O(\delta')$), and the *Binding* step takes linear time *w.r.t.*the number of frequent patterns. To obtain k-piece correlated patterns, all we need is one step of *PrefixSpan*, one step of *SuffixSpan*, and (k-1) *Binding* steps, so the overall computational complexity is $2 \cdot O(\delta) + O(\delta') + (k-1) \cdot O(\sigma) = O(\delta + \delta' + k\sigma)$.

Acceleration by Parallel Computing

As per Theorem 3, the computation of (k+1)-piece correlation score of a pattern only depends on the k-piece maximum cutting probability of its prefixes and the probability of its suffixes. For the *binding* step, these prefix probabilities are obtained from the sorted list generated by the *PrefixSpan* step (for 2-piece correlated patterns) or the previous *Binding* step (for (k+1)-piece correlated patterns); and these suffix probabilities are given by the *Suffix* step.



(a) the running time *w.r.t.* the size of database

(b) the running time *w.r.t.* the minimum support threshold

(c) the running time *w.r.t.* the minimum correlation threshold

Figure 3.2: PSBSpan V.S. PrefixSpan+

We split the first sorted list into several chunks, so that two patterns P and P' are always in the same chunk if P is a prefix of P'; and we split the second sorted list into the same chunks. Recall that the *suffix* step outputs the probability of a pattern's suffixes, along with that pattern. Thus, we can parallelize this approach and compute each chunk separately.

3.3 Experiments

In this section, we conduct our experiments on a real dataset [82], to show the performance of the *PSBSpan* algorithm. All the algorithms were implemented in Java (Eclipse Helio 2000) and the experiments were performed on a Windows 7 server with Intel Core2 Duo processors and 2GB of main memory.

Dataset. The Digital Bibliography and Library Project is a web accessible database of the bibliographic information of computer science publications. In this paper, we use a collection of DBLP articles [82] released by the ArnetMiner group of Tsinghua University, which contains $1,632,442$ publications and $1,741,170$ researchers. We consider therein $32,224$ papers published in prestigious conferences (*e.g.*, SIGKDD, SIGIR, SIGMOD, VLDB, CIKM, AAAI, *etc*) in the research areas of *database*, *data mining*, and *machine learning*.

Parameter. The parameters are set as: the database size $N = |\mathcal{SDB}| = 322,240$, the minimum

Measure	Top Ranked Patterns	
support	object oriented database	distributed database system
	database management system	relational database system
	object oriented system	data management system
	object database system	association rule mining
	support vector machine	oriented database system
	data base system	time series data
	object oriented database system	real tme database
all-confidence	object oriented database	association rule mining
	peer peer network	object oriented system
	nearest neighbor search	nearest neighbor query
	self organizing map	distributed database system
	concurrency control database	database management system
	relational database system	wireless sensor network
	real time database	mining association rule
lift	support vector machine	nonnegative matrix factorization
	reverse nearest neighbor	conditional random field
	named entity recognition	nearest neighbor moving
	latent dirichlet allocation	object oriented database
	nearest neighbor uncertain	singular value decomposition
	privacy preserving publishing	association rule mining
	continuous nearest neighbor	nearest neighbor search
cor	nonnegative matrix factorization	singular value decomposition
	conditional random field	named entity recognition
	aqualogic data service platform	latent dirichlet allocation
	association rule mining	join algorithm multiprocessor
	optimized rule numeric attribute	inductive logic programming
	reverse nearest neighbor	wireless data broadcast
	privacy preserving data publishing	message table content index

Table 3.5: Case Study: Comparing Four Measures

support threshold $min\_sup = 0.016\%$, and the minimum correlation threshold $min\_cor = 2.0$. The size of the vocabulary $\mathcal{E}$ is $13,942$. In Section 3.3.1 and Section 3.3.2, we explore varying the parameters and see how the performance is affected.

3.3.1 Frequent Patterns V.S. Correlated Patterns

As stated in Section 1, pattern mining with real datasets is difficult: when the minimum support threshold is set low, a huge number of patterns will usually be generated, a majority of which are redundant, uninformative or just random combinations of popular data objects. In this experiment,

we will prove that correlated patterns only make up a small set of all frequent patterns; and there is no dependency between a pattern's support and its correlation score.

In Figure 3.1, we plot the number of frequent patterns generated by *PrefixSpan* [66] and frequent correlated patterns extracted by our *PSBSpan* algorithm, while varying (i) the size of the database from $50K$ to $320K$ (Figure 3.1(a)), (ii) the minimum support threshold[2] from 0.006% to 0.028% (Figure 3.1(b)), and (iii) the minimum correlation threshold from 2.0 to 32.0 (Figure 3.1(c)).

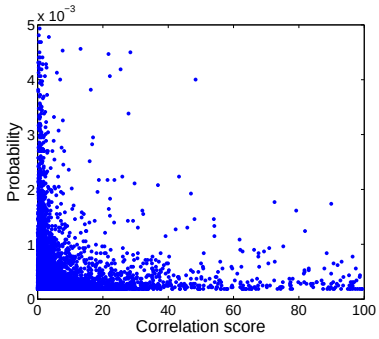


Figure 3.3: Probability V.S. Correlation

We could see that both curves show an approximate linear dependency on the database size N (Figure 3.1(a)), but exponentially increase (Figure 3.1(b)) when the minimum support threshold is set lower than 0.015%. Under the parameter settings in the two figures, the number of temporarily correlated patterns in the output of the *PrefixSpan* or *SuffixSpan* step (Section 3.2.2) ranges from 18.6% to 63.8%, and the ratio of the number of frequent correlated patterns versus the number of frequent patterns ranges from 14.1% to 52.8%. This proves that, in most cases, the majority of frequent patterns are not correlated.

The most interesting discovery that has never been discussed in the traditional setup of pattern mining tasks is the power law of the distribution of correlated patterns. It is shown in Figure 3.1(c) that the number of correlated patterns varies exponentially with the minimum correlation threshold. Thus, when we set the minimum correlation threshold high, 'truly' interesting patterns are only a

[2]The minimum support threshold is expressed as the ratio of the absolute count of supporting transactions over the total number of transactions.

tiny part of the huge pile of frequent patterns.

Furthermore, we extract patterns with probabilities no less than 0.016%, each of which is plotted as a dot in Figure 3.3 with the X- and Y- axeses as the correlation score and probability, respectively. We can see clearly that there is no dependency between the two measures. It also proves the necessity to add the correlation constraint to pattern mining tasks, since a frequent pattern may or may not be correlated.

3.3.2 Efficiency Evaluation

In this section, we will evaluate the mining efficiency of our *PSBSpan* algorithm

PrefixSpan+. As shown in Figure 3.1, the number of generated frequent patterns is as many as half a million, and creating indexes on these frequent patterns exceeds the availability of our primary memory. To avoid using indexes, we implemented an alternative method, called *PrefixSpan+*, as a baseline to compared with our *PSBSpan* algorithm. Generally speaking, *PrefixSpan+* is the same as the traditional *PrefixSpan* algorithm [66], but with additional correlation testing during the pattern growth based mining procedure. However, since the computation of the correlation score (Equation 3.2) of a sequence depends on the probability of any arbitrary sub-sequence, *PrefixSpan+* has no other choice but simply goes back to the original input database $\mathcal{SDB}$ to count the support of these sub-sequences by scanning the whole database $\mathcal{SDB}$.

In Figure 3.2, we show the running time (in second) of the two methods *PSBSpan* and *PrefixSpan+*, while varying the size of the input database (Figure 3.2(a)), the minimum support threshold (Figure 3.2(b)) and the minimum correlation threshold (Figure 3.2(b)). We could see that in all cases, *PSBSpan* significantly outperforms *PrefixSpan+*; and especially when the size of database is larger than $25K$ or the minimum support threshold is lower than 0.01%, the running time of *PrefixSpan+* becomes intolerable.

3.3.3 Case Study

There have been extensive studies on association measures, including χ^2, *lift*, *all-confidence*, *max-confidence*, *Kulcsynski*, *Cosine*, *etc.* In this experiment, we perform a case study to show the effectiveness of our method, not only compared with the mining result of *PrefixSpan* [66], but also with two measures *all-confidence* (Equation 3.7) and *lift* (Equation 3.1), which are popularly used in literature related to pattern mining and topic detection.

$$all\text{-}conf(e_1, e_2, \cdots, e_n) = \frac{Sup(e_1, e_2, \cdots, e_n)}{Max_{i=1..n}\{Sup(e_i)\}} \tag{3.7}$$

In Table 3.5, we list top-ranked sequential patterns (whose size is larger than two) according to four measures: *support* (see Definition 3.1.2), *all-confidence* (Equation 3.7), *lift* (Equation 3.1), and *cor* (Equation 3.2). We can see that patterns with highest support values are mostly random combinations of popular words; even though some phrases make sense in high level concepts, *e.g.*, *'database system'*, their useless duplicates may appear multiple times, such as, *'oriented database system'*, *'data base system'*, and *'object database system'*.

As we stated in Section 3.1.2, the measure *all-confidence* satisfies the Apriori property, *i.e.*, a super-pattern must have higher *all-confidence* score than its sub-patterns, which is obviously unreasonable in real situations. For instance, 'object oriented database system' is a meaningful phrase, but its useless sub-pattern 'object oriented system' is ranked higher (see the second row of Table 3.5).

The general ideas of the traditional measure *lift* defined on itemsets (Equation (3.1)) and the measure *cor* we proposed for sequential patterns (Equation (3.2)) are the same: the probability of a correlated pattern should be significantly larger than the joint probability of 'information units' that make up the correlated pattern. However, their difference relies on: the only possible information units considered by *lift* are single items appearing in the pattern, while the concept of information unit in *cor* is extended to any sub-pattern of the correlated pattern, so that *cor* considers more completely than *lift*. We expressed the extended concept by 'cutting' (Definition 3.1.3) and define

the *cor* measure by using cutting in Equation (3.2).

Due to the above reasons, 'nearest neighbor' is an interesting information unit, but 'nearest' and 'neighbor' separately are not. Using the *lift* measure, 'reverse nearest neighbor', 'nearest neighbor moving', 'nearest neighbor uncertain', and 'nearest neighbor search' all appear as highly ranked patterns (see the third row of Table 3.5), but only 'reverse nearest neighbor' is considered to be a truly correlated pattern by the *cor* measure (see the last row of Table 3.5).

3.4 Conclusion

In this chapter, I propose an algorithm for popular event detection, which is efficiently mining correlated frequent sequential patterns from a transactional sequence database extracted from social user-generated contents. To formally define the problem, we analyze 'good' and 'bad' properties for the selection of correlation measures. Moreover, we develop an efficient three-stage mining method, *Prefix-Suffix-Binding Span* (*PSBSpan*), based on an extension of pattern growth methodology. Experimental studies reveal that our mining method is able to discover 'truly' succinct and interesting popular events.

Algorithm 2 Binding(F_{pre}, F_{post}, min_cor)

Input:
the output file F_{pre} generated by the *PrefixSpan* step
the output file F_{post} generated by the *SuffixSpan* step, and
the minimum correlation threshold min_cor.

Set pointer p_1 to the begining of F_{pre}
Set pointer p_2 to the begining of F_{post}
while $p_1 \neq null$ and $p_2 \neq null$ **do**
  Let S_1 be the pattern pointed by p_1 in F_{pre}
  Let S_2 be the pattern pointed by p_2 in F_{post}
  if $S_1 \neq S_2$ **then**
    if $S_1 < S_2$ **then**
      $p_1 = p_1 + 1$
    else
      $p_2 = p_2 + 1$
    end if
  else
    Let $S = S_1 = S_2 = e_1 e_2 \cdots e_n$
    $Mcp = -\infty$
    for $i = 1 \rightarrow n$ **do**
      $p = Pr(e_1 e_2 \cdots e_i) Pr(e_{i+1} e_{i+2} \cdots e_n)$
      if $p > Mcp$ **then**
        $Mcp = p$
      end if
    end for
    output S, $Pr(S)$, Mcp_1, Mcp_2, $\cdots$, Mcp_{n-1}
    $Mcp_n = Mcp$
    $p_1 = p_1 + 1$
    $p_2 = p_2 + 1$
  end if
end while
Set pointer $p1$ to the ending position of the output
Set pointer $S_2 =$ null.
while $p_1 \neq$ null **do**
  Let S_1 be the pattern pointed by p_1
  if 2-$Cor(S_1)$ ¡ min_cor and $S_1 \subset S_2$ **then**
    delete S_1
  end if
  if 2-$Cor(S_1) \geq min\_cor$ **then**
    S_2 = S_1
  end if
  $p_1 = p_1 - 1$
end while

Chapter 4

Event Popularity Tracking in Social Communities

In the section, I propose a novel and principled probabilistic model (called *PET*) for *Popular Events Tracking* in a time-variant social community that consists of dynamic textual and structural information. Specifically, *PET* takes (i) a stream of document collections, (ii) a stream of network structures, and (iii) a popular event, as the input, and generates analysis on the event in both aspects of popularity and content evolution.

PET leverages a Gibbs Random Field to model the interest of users, depending on their historical status as well as the influence from their social connections. A topic model is designed to explain the generation of text data given the interest of a user in an event. The Gibbs Random Field and the topic model thus interplay by regularizing each other. The tasks of tracking popular events are thus cast as an optimization problem aiming at the inference of a joint distribution that consolidates all of historic, textual, and structural features.

We show that *PET* is motivated by and well reflects the existing observations and findings about information diffusion in social networks and the topic burstiness in text. *PET* is well connected to two classical models [43, 62], which are proved to be special cases of *PET* under certain situations. Empirical experiments on two different online communities *Twitter* and *DBLP* show that our approach is effective and outperforms various baselines.

Organization. Section 4.1 formally defines the problem of popular event tracking, as the solution of which a unified probabilistic model (called *PET*) is proposed in Section 4.2. Section 4.3 discusses several other issues related to *PET*, including an extension model, the connection with two classical models [43, 62] in literature, and the analysis on time complexity. We then present experiments and results on two real datasets *Twitter* and *DBLP* in Section 4.4.

4.1 Problem Formulation

In this section, we formally define the related concepts and the task of popular event tracking in social communities. Let us begin with defining a few key concepts as follows.

Definition 4.1.1. **Network Stream**. Let $G = \{G_1, G_2, \cdots, G_T\}$ be a stream of network structures, where G_k is a snapshot of a general network G at time k for $k \in [1, 2, \cdots, T]$. $G_k = (V_k, E_k)$, where V_k is a set of vertices and E_k is a set of edges among V_k. In a social network, a vertex corresponds to a user. An edge $e = (i, j) \in E_k$ stands for a connection (or a tie) between vertices i and j. We define $g_k(i, j)$ as the strength of the tie (i, j) at time t_k. *W.l.o.g*, we define G_k as a complete graph but allow $g_k(i, j)$ to be any non-negative real value, *i.e.*, $g_k(i, j) = 0$ if there is no tie between vertices i and j. Note G_k can be either undirected (*e.g.*, co-authorship) or directed (*e.g.*, follow-followee relationship).

Definition 4.1.2. **Document Stream**. Let $D = \{D_1, D_2, \cdots, D_T\}$ be a stream of document collections, where D_k is the set of documents published between time $(k-1)$ and k. We further denote $D_k = \{d_{k,1}, d_{k,2}, \cdots, d_{k,N}\}$, where $d_{k,i}$ is the text document(s) associated with the node $v_{k,i}$ in G_k. Document $d_{k,i}$ is represented by a bag of words from a fixed vocabulary W. That is, $d_{k,i} = \{c(d_{k,i}, w)\}_{w \in W}$, where $c(d_{k,i}, w)$ denotes the number of occurrences of word w in $d_{k,i}$.

Definition 4.1.3. **Topic**. We present a semantically coherent topic θ as a multinomial distribution of words, *i.e.*, $\{p(w|\theta)\}_{w \in W}$ with the constraint $\sum_{w \in W} p(w|\theta) = 1$. We allow a topic to have different versions over time, *i.e.*, the version of θ at time k is denoted as θ_k.

Definition 4.1.4. **Event**. We define a general event E as a stream of topics $\Theta^E = \{\theta_0^E, \theta_1^E, \theta_2^E, \cdots, \theta_T^E\}$. We call θ_0^E the primitive topic of the event, which is independent of the network. θ_0^E can either be specified by the users or be automatically discovered by an event detection algorithm [24, 65, 96]. θ_k^E corresponds to the version of θ_0^E at time k. θ_k^E is dependent of the network, which indicates the major aspects of the event in network G_k. Altogether Θ^E represents the origin and evolution of the contents of the event E over time. We use $\Theta, \theta_0, \theta_k$ to denote $\Theta^E, \theta_0^E, \theta_k^E$ when

there is no ambiguity.

Definition 4.1.5. **Interest**. For a particular event, at each time point k, we assume each node v_i in G_k has a certain level of interest in the event. We model such level of interest as a real value $h_k(i) \in [0, 1]$, and denote the set of interest values for all vertices in G_k as H_k, *i.e.*, $H_k = \{h_k(i)\}_{v_k(i) \in V_k}$. Note that one can also define h with a set of discrete levels.

Based on the definitions above, we can define the event-related information in a social community as 1) an *observed* stream of network structures; 2) an *observed* stream of text documents; 3) a *latent* stream of topics about the event; and 4) a *latent* stream of interests. We illustrate these concepts with two real world social communities *Twitter* and *DBLP*.

Example 4.1.1. *For Twitter (a micro-blogging network), we extract a collection of N users and all posts published by these users in a range of T days. A time point k is defined as the k^{th} day in the time range. $d_{k,i}$ is the document obtained by concatenating all tweets published by user i on the day k. The edge weight $g_k(i, j)$ is an estimation of how much user i is influenced by user j on day k,* e.g., *$g_k(i, j)$ could be defined as the number of i's tweets that follow j from day 1 to day k. Here, G_k is a directed graph.*

Example 4.1.2. *For DBLP (a bibliographic network), we retrieve N authors and all publications of these authors in T years. A time point k corresponds to the k^{th} year. $d_{k,i}$ is the concatenation of titles of author i's papers in year k. The network G_k is created among these authors according to their co-author relationship. $g_k(i, j)$ is defined as the number of papers co-authored by author i and j in year k, so here G_k is an undirected graph.*

With the definitions of related concepts, we can now formally define the major tasks in the problem of popular event tracking in social communities. Given the input of network stream G, document stream D and the primitive topic of an event θ_0, the tasks include:

Task 1: Popularity Tracking. Formally, we want to infer the latent stream of interests, *i.e.*, H_k at each time point t_k during the tracking period. The H_k values can not only indicate the overall

popularity trend of the event, but also provide much richer information about how the interest develops, evolves, and spreads on the network.

Task 2: Topic Tracking. Formally, we want to infer the latent stream of topics about the event Θ over time. An event starts from its primitive form θ_0, and while it is developing, the major aspects of the event may shift substantially over time. By inferring the stream of topics, we expect to keep track of the new development about the event, understand its evolution, and identify the most attentive aspect of the event to the community over time, *etc.*

Tracking the popular events in a social community is important and challenging in many ways. To track the popularity of events on the network, we should figure out how the interest of each individual is influenced by its social connections, and then develop reasonable models to simulate the formulation and diffusion of the interest on the networks. To track the content evolution of the event, we should make sure the topics we track should be always relevant to the event, and more importantly, reflect the current interest of individuals on the network. This requires us to propose a unified model that takes consideration of interest diffusion, network structure and textual contents at the same time.

It is also worth mentioning that in this work we focus on event tracking, not detection, since the primitive event topic θ_0^E is considered as input. We have observed that in general events could be well described by several keywords, *e.g.*, 'Avatar', 'Tiger Woods Accident', so it is feasible for users to provide the primitive event topic. Indeed, our approach could be combined with any existing event detection algorithms [24, 97] that automatically discover the bursty keywords or topics either from the same network or other sources, *e.g.*, news articles or searching engine, then our system will track the events on the focused network.

4.2 Event Tracking Models

In this section, we present a novel probabilistic model, **PET**, for tracking popular events in social communities. By considering both the evolution of textual documents and the evolution of network

structures, our model can capture the popularity and topic evolution of events in a unified process.

As discussed in Section 4.1, a reasonable model of tracking popular events in a social community should not only capture the diffusion of information through the network, but also the burstiness of interests and the generation of contents. What factors should *PET* consider? What existing observations in social networks and text mining could *PET* utilize? Before formally introducing the model, we first explain several key observations that motivate the model:

Observation 1. **Interest and Connections.** It has been shown in the literature of information diffusion that the behavior of a social actor (*e.g.*, $v_i \in G$ in out input setting) is usually influenced by its friends [49], especially friends that have stronger ties with v_i [11]. We may expect that the cascade behavior also applies to the interest in an event. Moreover, $v_i's$ friends have an even stronger social influence on the interest of v_i's if v_i's friends have similar interests or v_i's friends with the same interest are strongly connected [6]. On the other hand, the study of homophily efforts has shown that people with similar interests are more likely to become connected [2, 23].

Observation 2. **Interest and History.** The behavior of each individual should be generally consistent over time [43], thus present a strong 'personalized' pattern. This also means interest towards certain events, at most time, should not change dramatically within a short time. When there is a bursting pattern of the interest at time k, it's more likely to remain at a high level at time $(k+1)$ [43].

Observation 3. **Content and Interest**. When an individual v_i has a higher level of interest in an event, the content she generates should be more likely to be related to the event [73]. On the other hand, when we find v_i writes more about the event, we can assume she is more interested in the event.

We expect these intuitions and observations be helpful in designing the probabilistic model.

4.2.1 The General Model

Now, at time k, we already know the network stream G_1, G_2, $\cdots$, G_k and the document stream D_1, D_2, $\cdots$, G_k. Let us assume that we have also known the previous interest values H_1, H_2, $\cdots$, H_{k-1}. We want to infer the current interest value H_k and topics θ_k on the network. We may further make an Markovian simplification that the current interest status only depends on the previous status, *i.e.*, H_{k-1}. So formally, the task is cast as the inference of the posterior of H_k and θ_k as

$$P(H_k, \theta_k | G_k, D_k, H_{k-1}) \tag{4.1}$$

Based on the intuitions and observations, we know H_k depends on the network structure G_k (*i.e.*, Observation 1) as well as the history H_{k-1} (*i.e.*, Observation 2). We also know that the topic θ_k depends on the current interest status H_k (*i.e.*, Observation 3). We can then introduce two reasonable independent assumptions:

1. Given the current network structure G_k and the previous interest status H_{k-1}, the current interest status H_k is independent of the document collection D_k. The intuition is that people first become interested in the event and therefore generate discussions on it, *i.e.*, H_k influences D_k but not reversely.

2. Given the current interest status H_k and the document collection D_k, the current topic model θ_k is independent of the network structure G_k and the previous interest status H_{k-1}. The intuition is that once the author v_i has developed an interest in the event, the content she writes will only depend on the event itself and the level of the interest.

With the above two assumptions, our object becomes to infer:

$$P(H_k, \theta_k | G_k, D_k, H_{k-1}) = P(H_k | G_k, H_{k-1}) \cdot P(\theta_k | H_k, D_k) \tag{4.2}$$

We denote the left component, *i.e.*, $P(H_k | G_k, H_{k-1})$, in Equation 4.2 as the interest model and

the right component, $P(\theta_k|H_k, D_k)$, as the topic model. In the interest model, we propose a multivariate Gibbs Random Field [51] to model the dependency among individuals and the influence of past status (Section 4.2.2); in the topic model, a mixture model [95] is designed to extract the topic snapshot of the event (Section 4.2.3). Finally, the inference of the combined model is discussed in Section 4.2.4.

4.2.2 The Interest Model

Let us first briefly introduce the Gibbs Random Field.

Gibbs Random Field. Given a graph $G = \{V, E\}$, a family of random variables $F = \{F_i\}_{i=1}^N$ is said to be a Gibbs Random Field *w.r.t.* G if and only if its configuration, f, follows a Gibbs distribution that takes the form

$$P(f) = Z^{-1} \times e^{-\frac{1}{\lambda_T}U(f)},$$

where $Z = \sum_{f\in\mathbb{F}} P(f)$ is a normalizing constant called the *partition function*, λ_T is a constant called the *temperature*, and the *energy function* $U(f) = \sum_c V_c(f)$ is a sum of *clique potentials* $V_c(f)$ over all possible cliques c.

In our model, the interest status H_k is a family of random variables defined on graph G_k, and we give a configuration of H_k that follows a Gibbs distribution:

$$P(H_k|G_k, H_{k-1}) = Z^{-1} \times e^{-\frac{1}{\lambda_T}U(H_k)}$$

For the energy function $U(H_k)$, we specifically define two kinds of clique potential functions, while set all other potentials to 0, *i.e.*,

$$U(H_k) = \sum_{i=1}^{N} V_i(h_k(i)) + \sum_{i=1}^{N} V_i'(h_k(i), h_k(-i)) \tag{4.3}$$

In Equation 4.3, $-i$ refers to the set of all vertices except i. Note $h_k(i)$ itself is a size-1 clique

in G_k, and $\{h_k(i), h_k(-i)\}$ simply equals to G_k, which is also a clique. Hence, Equation 4.3 is a valid Gibbs Random Field.

We then define $V_i(h_k(i))$ as the transition energy of node i from its last status $h_{k-1}(i)$ to current status $h_k(i)$:

$$V_i(h_k(i)) = (h_k(i) - h_{k-1}(i))^2, \forall i \in [1..N]$$

This definition is mainly motivated by our Observation 2: by minimizing this transition cost we would like the interest values to be generally consistent over time.

The other kind of potential function $V'_i(h_k(i), h_k(-i))$ gives penalty for the difference between the interest of i and its expected value:

$$V'_i(h_k(i), h_k(-i)) = \lambda_{k,i}(h_k(i) - h'_k(i)))^2, \forall i \in [1..N], \tag{4.4}$$

where $h'_k(i)$ is the expectation of $h_k(i)$ estimated from i's neighbors $n(i)$ as

$$h'_k(i) = \frac{\sum_{j \in n(i)} g_k(i,j) \cdot h_{k-1}(j)}{\sum_{j \in n(i)} g_k(i,j)}$$

We can see that the design of this cost function is motivated by our Observation 1, which well captures the intuitions in information diffusion: i's current interest is influenced by i's connections, and a stronger tie (*i.e.*, higher $g_k(i,j)$) brings a larger impact.

Moreover, in Equation 4.4, $\lambda_{k,i}$ is a weight that represents overall how much we trust the 'influence from friends', *i.e.*,

$$\lambda_{k,i} = \lambda_A \cdot \left(\sum_{j \in n(i)} g_k(i,j) \right) \cdot (1 - \xi(i)) \,,$$

where λ_A is a constant and $\xi(i)$ is the harmonic function [100] defined on the neighbor graph of i

as:

$$\xi(i) = \frac{\sum_{j_1,j_2 \in n(i), j_1 \neq j_2} g_k(j_1, j_2) \cdot (h_{k-1}(j_1) - h_{k-1}(j_2))^2}{\sum_{j_1,j_2 \in n(i), j_1 \neq j_2} g_k(j_1, j_2)}$$

The definition of $\lambda_{k,i}$ well captures another intuition in our Observation 1: when i's neighbors have a higher agreement on the interest value, the harmonic function becomes smaller, thus results in larger $\lambda_{k,i}$. For special conditions, *i.e.*, $\sum_{j \in n(i)} g_k(i, j) = 0$, we can simply set $h'_k(i)$ to an arbitrary value and set $\lambda_{k,i}$ to zero.

To sum up, the posterior of interest status $P(H_k|G_k, H_{k-1})$ is modeled as a Gibbs Random Field on the network G_k. Several potential functions are designed in order to let the interest value of each individual be close to the past status and the 'agreement' of the neighbors.

4.2.3 The Topic Model

Now we consider the topic component (*i.e.*, $P(\theta_k|H_k, D_k)$) in Equation 4.2. In our model, we consider each document $d_{i,k}$ in the collection D_k is generated from a mixture of two multinomial component models. One component model is a background model θ^B and the other is the latent event topic model θ_k that we want to estimate. The idea is to model the common (non-discriminative) words in D_k with θ_k^B so that the event topic model θ_k would attract more discriminative and meaningful words that describe the target event.

The generation process is as follows: to write a word in document $d_{i,k}$, one first chooses between the event topic mode (*i.e.*, θ_k) and the background model (*i.e.*, θ_k^B), with probability $p(\theta_k|d_{k,i})$ and $p(\theta^B|d_{k,i})$, respectively, with the constraint $p(\theta_k|d_{k,i}) + p(\theta^B|d_{k,i}) = 1$. Once the topic is selected, one samples a word from either the event topic model or the background model. Different from the traditional mixture language models [32, 95], where the topic distribution of each document is either predefined or solely estimated from the text data, in our model we use the interest value $h_k(i)$, a real value in $[0, 1]$, as the probability of choosing the event topic at node i, *i.e.*, $p(\theta_k|d_{k,i}) = h_k(i)$. This is reasonable according to our Observation 3: a higher interest of

v_i in the event should result in a higher proportion of the event covered in by v_i. Moreover, as explained in the interest model, $h_k(i)$ could capture the historical interest status and relationship on the network, which implicitly influences the topic model. And modeling the joint distribution with both components would allow the topics and popularity of the event to mutually influence each other over time.

Formally, the probability of generating word w in $d_{k,i}$ is:

$$p(w|d_{k,i}) = h_k(i)p(w|\theta_k) + (1 - h_k(i))p(w|\theta^B) \tag{4.5}$$

We assume $p(w|\theta^B)$ does not change over time, which can be simply estimated by the maximum likelihood estimator using the entire document stream. So the likelihood of the document collection D_k is given as:

$$P(D_{k,i}|H_k, \theta_k) = \prod_{i=1} \prod_{w \in W} p(w|d_{k,i})^{c(d_{k,i},w)},$$

where $c(d_{k,i}, w)$ is the number of occurrences of w in $d_{k,i}$.

We further define a conjugate Dirichlet prior of the event topic θ_k: $Dir(\{1+\mu_E p(w|\theta_0)\}_{w \in W})$, to incorporate the primitive event topic, which servers as the prior knowledge of the event. By doing this, we regularize the topics so that they do not shift from the event. μ_E is the weight indicating how much we rely on the prior. Formally,

$$P(\theta_k|H_k) = P(\theta_k) \propto \prod_{w \in W} p(w|\theta_k)^{\mu_E p(w|\theta_0)}$$

By considering D_k as the observation (*i.e.*, $P(D_k|H_k)$ is a constant), the posterior of topics θ_k is given as:

$$P(\theta_k|H_k, D_k) \propto P(D_k|H_k, \theta_k)P(\theta_k|H_k)$$

4.2.4 Parameter Estimation

Given our model defined in Equation 4.2, we can fit the model to the data and estimate the parameters using a Maximum A Posterior estimator. That is:

$$\Lambda^* = \underset{\Lambda}{\operatorname{argmax}}\, p(C|\Lambda)p(\Lambda)$$

where Λ has the interest values $h_k(i)$ and word distribution in the topic models θ_k. The hidden variable in our model is $z_{d_{k,i},w}$, indicating which topic (*i.e.*, θ_k or θ^B) is selected to generate word w in document $d_{k,i}$.

The Expectation Maximization (EM) algorithm [59] can be applied to estimate the parameters efficiently. In the E-step, it computes the expectation of the hidden variables; and in the M-step, it updates parameters Λ to maximize the object function given above.

Specifically, in the E-step we have:

$$p^{(n)}_{d_k,i}(z_w = \theta_k) = \frac{h_k^{(n-1)}(i)p^{(n-1)}(w|\theta_k)}{h_k^{(n-1)}(i)p^{(n-1)}(w|\theta_k) + (h_k^B)^{(n-1)}(i)p^{(n-1)}(w|\theta^B)}$$

$$p^{(n)}_{d_k,i}(z_w = \theta^B) = \frac{(h_k^B)^{(n-1)}(i)p^{(n-1)}(w|\theta^B)}{h_k^{(n-1)}(i)p^{(n-1)}(w|\theta_k) + (h_k^B)^{(n-1)}(i)p^{(n-1)}(w|\theta^B)},$$

where $h_k^B(i) = 1 - h_k(i)$ for the convenience of express.

In the M-step, given the expectation of the hidden variables, the object function we want to

maximize is

$$
\begin{aligned}
E_{\Lambda^{(n-1)}}\{\log p(C|\Lambda)p(\Lambda)\} \propto & -\log Z - \frac{1}{\lambda_T}\sum_{i=1}^{N}\left((h_k(i)-h_{k-1}(i))^2 + \lambda_{k,i}(h_k(i)-h'_k(i))^2\right) \\
& + \sum_{i=1}^{N}\sum_{w\in W} c(d_{k,i},w)p^{(n)}_{d_{k,i}}(z_w=\theta_k)\log(h_k(i)p(w|\theta_k)) \\
& + \sum_{i=1}^{N}\sum_{w\in W} c(d_{k,i},w)p^{(n)}_{d_{k,i}}(z_w=\theta^B)\log(h^B_k(i)p(w|\theta^B)) \\
& + \sum_{w\in W}\mu_E p(w|\theta^E_0)\log(p(w|\theta^E_k))
\end{aligned}
$$

By integrating a few Lagrange multipliers [59], we can get:

$$
p^{(n)}(w|\theta_k) = \frac{\sum_{i=1}^{N} c(d_{k,i},w)p^{(n)}_{d_{k,i}}(z_w=\theta_k) + \mu_E p(w|\theta_0)}{\sum_{w'\in W}\sum_{i=1}^{N} c(d_{k,i},w')p^{(n)}_{d_{k,i}}(z_{w'}=\theta_k) + \mu_E}
$$

The inference of $h_k(i)$ boils down to solve:

$$
\alpha h_k(i) - \beta - \frac{\gamma}{h_k(i)} - \frac{\delta}{h_k(i)-1} = 0 \tag{4.6}
$$

where

$$
\begin{aligned}
\alpha &= \frac{2}{\lambda_T}(1+\lambda_{k,i}), \\
\beta &= \frac{2}{\lambda_T}(h_{k-1}(i) + \lambda_{k,i}h'_k(i)), \\
\gamma &= \sum_{w\in W} c(d_{k,i},w)p^{(n)}_{d_{k,i}}(z_w=\theta^E_k), \\
\delta &= \sum_{w\in W} c(d_{k,i},w)p^{(n)}_{d_{k,i}}(z_w=\theta^B_k).
\end{aligned}
$$

Equation 4.6 can be solved according to two cases:

1. When $\sum_{w \in W} c(d_{k,i}, w) = 0$, *i.e.*, the document d_i is empty at time point t^k. Then $\gamma = 0$ and $\delta = 0$, so that $h_k(i)$ only depends on the information from the past status and neighbors:

$$h_k(i) = \frac{\beta}{\alpha} = \frac{h_{k-1}(i) + \lambda_{k,i} h'_k(i)}{1 + \lambda_{k,i}} \tag{4.7}$$

2. When $\sum_{w \in W} c(d_{k,i}, w) > 0$, Equation 4.6 is equivalent to a one variable cubic function:

$$\alpha h_k(i)^3 - (\alpha + \beta) h_k(i)^2 + (\beta - \gamma - \delta) h_k(i) + \gamma = 0 \tag{4.8}$$

   Any efficient root searching approaches for cubic functions can be applied to find the feasible $h_k(i)$ that satisfies Equation 4.8. Denote the left of the equation as $f(h_k(i))$. Then $f(-\infty) = -\infty$, $f(+\infty) = +\infty$, $f(0) = \gamma > 0$, $f(1) = -\delta < 0$. It is easy to show there exists exact one root in $(0, 1)$, and therefore the solution for $h_k(i)$ is guaranteed to be found.

To sum up, motivated by three key intuitions, *PET* casts the task of tracking popular events as an optimization problem aiming at the inference of an object function consolidating historic, textual, and structural features (Section 4.2.1). Furthermore, the object is decomposed into a interest model and a topic model, where the former one utilizes a Gibbs Random Field to model the interest levels of users (Section 4.2.2), depending on their historical status and social connections, and the latter one explains the generation of text data given the interest of a user (Section 4.2.3). Finally, the two components interplay by regularizing each other, and the parameters of the model are estimated by the Expectation Maximization algorithm (Section 4.2.4).

4.3 Discussions

In this section, we discuss several other issues, including extending the *PET* model to the one of tracking multi-events simultaneously (Section 4.3.1), its connection with two famous existing models (Section 4.3.2), and finally the analysis of the time complexity (Section 4.3.3).

4.3.1 Extended for Multiple Events Tracking

In the topic model, the event model θ_k^E (we abbreviate θ_k^E as θ_k in Section 4.2) and the background model θ^B are used to generate words in documents. The idea is to model common words with θ^B, so that θ_k^E could attract more discriminative and meaningful words related to the event. However, when multiple events happened at the same time over the social community, some none-common words from other events may also be absorbed by θ_k^E and thus the effectiveness is reduced.

Because of such concern, it is nature to extend *PET* to the one of tracking multi-events simultaneously. We re-define the problem as: given a document stream $D = \{D_1, D_2, \cdots, D_T\}$, a network stream $G = \{G_1, G_2, \cdots, G_T\}$ and a set of events $E = \{E_1, E_2, \cdots, E_M\}$ with $\theta_0^{E_p}$ as the prior knowledge for E_p, we want to infer the latent interest status $H_k^{E_p}$ and the latent topic $\theta_k^{E_p}$ for each event E_p at each time point k. For the convenience of express, we abbreviate $\{H_k^{E_1}, H_k^{E_2}, \cdots, H_k^{E_p}\}$ and $\{\theta_k^{E_1}, \theta_k^{E_2}, \cdots, \theta_k^{E_p}\}$ as H_k^E and θ_k^E, respectively. The object function for optimization that was originally displayed by Equation 4.2 therefore becomes

$$P(H_k^E, \theta_k^E | G_k, D_k, H_{k-1}^E) = P(H_k^E | G_k, H_{k-1}^E) \cdot P(\Theta_k^E | H_k^E, D_k),$$

where $P(H_k^E | G_k, H_{k-1}^E)$ and $P(\Theta_k^E | H_k^E, D_k)$ are the interest and the topic model, respectively.

For the interest model, the family of random variables in the Gibbs Random Field is extended from the interest status of one event to the ones of multiple events, and thus we give a configuration that follows a Gibbs distribution as:

$$P(H_k^E | G_k, H_{k-1}^E) = Z^{-1} \times e^{-\frac{1}{\lambda_T} U(H_k^E)}$$

The energy function becomes $U(H_k^E) = \sum_{E_p \in E} U(H_k^{E_p})$, where $U(H_k^{E_p})$ remains the same as defined by Equation 4.3 for each event $E_p \in E$.

For the topic model, each document $d_{k,i}$ is considered to have a potentially different set of mixing weights which captures the coverage of different models, including any event model and

the background model. To write a word in document $d_{k,i}$, one first chooses a model θ with the probability $p(\theta|d_{k,i})$, and then samples a word from the selected model θ with the probability $p(w|\theta)$ with the constraint $\sum_\theta p(\theta) = 1$. Motivated by Observation 3, we have $p(\theta_k^{E_p}|d_{k,i}) = h_k^{E_p}(i)$, and $p(\theta^B|d_{k,i}) = 1 - \sum_{E_p \in E} h_k^{E_p}(i)$. Formally, the posterior of topics θ_k^E is modified as:

$$P(\theta_k^E|H_k^E, D_k) \propto P(D_k|H_k^E, \theta_k^E)P(\theta_k^E|H_k^E)$$

The first component $P(D_k|H_k^E, \theta_k^E)$ is the likelihood of the document collection D_k, defined as

$$P(D_k|H_k^E, \theta_k^E) = \prod_{i=1} \prod_{w \in W} p(w|d_{k,i})^{c(w|d_{k,i})},$$

where the probability of generating word w in $d_{k,i}$ is:

$$p(w|d_{k,i}) = \sum_{E_p \in E} h_k^{E_p}(i)p(w|\theta_k^{E_p}) + h_k^B(i)p(w|\theta_k^B)$$

The second component $P(\theta_k^E|H_k^E)$ is the Dirichlet Prior defined as:

$$P(\theta_k^E|H_k^E) = \prod_{p=1}^{M} P(\theta_k^{E_p}) \propto \prod_{p=1}^{M} \prod_{w \in W} p(w|\theta_k^{E_p})^{\mu_{E_p} p(w|\theta_0^{E_p})}$$

Finally, the Expectation Maximization algorithm is applied to estimate H_k^E and θ_k^E. In the E-step, we have:

$$p_{d_{k,i}}^{(n)}(z_w = \theta_k^{E_p}) = \frac{(h_k^{E_p}(i))^{(n-1)} p^{(n-1)}(w|\theta_k^{E_p})}{\sum_{p=1}^{M} (h_k^{E_p}(i))^{(n-1)} p^{(n-1)}(w|\theta_k^{E_p}) + (h_k^B(i))^{(n-1)}) p(w|\theta_k^B)}$$

$$p_{d_{k,i}}^{(n)}(z_w = \theta_k^B) = \frac{(h_k^B(i))^{(n-1)} p^{(n-1)}(w|\theta^B)}{\sum_{p=1}^{M} (h_k^{E_p}(i))^{(n-1)} p^{(n-1)}(w|\theta_k^{E_p}) + (h_k^B(i))^{(n-1)}) p(w|\theta_k^B)}$$

In the M-step, the object function for maximization is:

$$
\begin{aligned}
& E_{\Lambda^{(n-1)}}\{\log p(C|\Lambda)p(\Lambda)\} \\
\propto & -\log Z - \frac{1}{\lambda_T}\sum_{p=1}^{M}\sum_{i=1}^{N}(h_k^{E_p}(i) - h_{k-1}^{E_p}(i))^2 \\
& - \frac{\lambda_{k,i}^{E_p}}{\lambda_T}\sum_{p=1}^{M}\sum_{i=1}^{N}(h_k^{E_p}(i) - h_k^{'E_p}(i))^2 \\
& + \sum_{p=1}^{M}\sum_{i=1}^{N}\sum_{w\in W} c(d_{k,i},w)p_{d_{k,i}}^{(n)}(z_w = \theta_k^{E_p})\log(h_k^{E_p}(i)p(w|\theta_k^{E_p})) \\
& + \sum_{i=1}^{N}\sum_{w\in W} c(d_{k,i},w)p_{d_{k,i}}^{(n)}(z_w = \theta^{B})\log(h_k^{B}(i)p(w|\theta^{B})) \\
& + \sum_{p=1}^{M}\sum_{w\in W}\mu_E p(w|\theta_0^{E_p})\log(p(w|\theta_k^{E_p}))
\end{aligned}
$$

By integrating Lagrange multipliers, we get:

$$
p^{(n)}(w|\theta_k^E) = \frac{\sum_{p=1}^{M}\sum_{i=1}^{N} c(d_{k,i},w)p_{d_{k,i}}^{(n)}(z_w = \theta_k^{E_p}) + \mu_E p(w|\theta_0^{E_p})}{\sum_{p=1}^{M}\sum_{i=1}^{N}\sum_{w'\in W} c(d_{k,i},w')p_{d_{k,i}}^{(n)}(z_{w'} = \theta_k^{E_p}) + \mu_E}
$$

The inference of $h_k^{E_p}(i)$ boils down to solve:

$$
\alpha h_k^{E_p}(i) - \beta - \frac{\gamma}{h_k^{E_p}(i)} - \frac{\delta}{h_k^{E_p}(i) - \sigma} = 0 \tag{4.9}
$$

where

$$\alpha = \frac{2}{\lambda_T}(1 + \lambda_{k,i}),$$
$$\beta = \frac{2}{\lambda_T}(h_{k-1}^{E_p}(i) + \lambda_{k,i} h_k^{'E_p}(i)),$$
$$\gamma = \sum_{w \in W} c(d_{k,i}, w) p_{d_{k,i}}^{(n)}(z_w = \theta_k^{E_p}),$$
$$\delta = \sum_{w \in W} c(d_{k,i}, w) p_{d_{k,i}}^{(n)}(z_w = \theta_k^{B}),$$
$$\sigma = 1 - \sum_{p=1, p \neq i}^{M} (h_k^{E_p}(i)) \approx 1 - \sum_{p=1, p \neq i}^{M} (h_k^{E_p}(i))^{(n-1)}.$$

The solution of Equation 4.9 is similar as Section 4.2.4.

4.3.2 The Connection with Existing Models

We have presented the model and the inference of *PET*. Although it is a novel probabilistic model, it is well connected to existing models in literature. In this section, we describe two famous existing models of word burstiness and network diffusion, and show that both of them are special cases of *PET* under certain situations: when the network effect in *PET* is omitted, it is well connected to the first model (Section 4.3.2); on the other hand, when the topic effect of *PET* is omitted, it is well connected to the second model (Section 4.3.2).

The State Automation Model

The first is a state automation model proposed by Kleinberg, *et al.* in [43] in the context of detecting bursting activities in an email stream. It is an HMM-like model which assumes the intervals between messages depend on the hidden 'bursty' states. We look at a variation of this model which matches our counting data: taking a sequence of counting of messages $X = \{x_1, x_2, \cdots, x_T\}$ as the observation, we aims at estimating a sequence of hidden states $\Lambda = \{\lambda_1, \lambda_2, \cdots,$

$\lambda_T\}$ that maximize the likelihood as below:

$$P(X|\Lambda) = P(x_1|\lambda_1) \cdot \prod_{k=2}^{T}(P(\lambda_k|\lambda_{k-1})P(x_k|\lambda_k)) \tag{4.10}$$

where $P(\lambda_k|\lambda_{k-1})$ is the transition probability. Instead of the exponential density function in [43], we define the probability distribution $P(x_k|\lambda_k)$ by a Poisson distribution as

$$P(x_k|\lambda_k) \propto \frac{\lambda_k^{x_k} e^{-\lambda_k}}{x_k!},$$

since Poisson is much more natural to model word counts [19]. Under the condition that transition probabilities is a constant, the maximum likelihood estimator gives λ_k based on:

$$\lambda_k^* = \underset{\lambda_k}{\operatorname{argmax}} \lambda_t^{x_k} e^{-\lambda_k}$$

Now we show this is a special case of *PET* by setting a few constraints:

1. We have $\lambda_T = \infty$ to eliminate the efforts of network and history. We also combine all nodes in G_k as a pseudo node $v_k(0)$, so that $d_{k,0}$ and $h_k(0)$ correspond to the document and the interest value for the whole network.

2. We have $\mu_E = 0$ to make the Dirichlet prior a constant, *i.e.*, $P(\theta_k|H_k) = 1$.

3. We assume there are only two pseudo words in the vocabulary, *i.e.*, the event word w_1 and the background word w_2, that $p(w_1|\theta_k) = 1$ and $p(w_2|\theta^B) = 1$.

Then the above model is transformed to a binomial distribution as:

$$P(\theta_k|H_k, D_k) \propto h_k(0)^{c(d_{k,0},w_1)} \times (1 - h_k(0))^{c(d_{k,0},w_2)}$$

We know that a Poisson distribution can be described as a limiting case of a binomial distribution. Specifically when total number of words $|d_{k,0}|$ is sufficiently large, the Poisson distribution

above is approximately equivalent to the binomial distribution in our topic model, and we have:

$$\lambda_k \approx n \cdot h_k(0)$$

This well connects *PET* with the state automation model.

The Contagion Model

Let us look at another classical model in the context of information diffusion, *i.e.*, the contagion model introduced in [62]. The general idea is that a person becomes infected (corresponding to the case that a person is interested in an event) if the number of its infected friends in the last time point is above a threshold. Let us simplify *PET* as follows:

1. $c(d_{k,i}, w) = 0$ for any node i and word w, *i.e.*, the part of the topic model disappears.
2. $g_k(i, j) = 1$ if v_i is influenced by v_j and otherwise 0.
3. $\lambda_A = \infty$ (so that $\lambda_{k,i} = \infty$), *i.e.*, the influence of neighbors dominates the one of history.

According to Equation 4.7, we have

$$h_k(i) = \lim_{\lambda_{k,i} \to \infty} \frac{h_{k-1}(i) + \lambda_{k,i} h'_k(i)}{1 + \lambda_{k,i}} = h'_k(i),$$

where $h'_k(i)$ equals to the ratio of infected friends of v_i at the last time point. If we set $h_k(i)$ to 1 only when $h'_k(i)$ is larger than a threshold, otherwise $h_k(i)$ remains 0, this is equivalent to the contagion model.

4.3.3 Complexity Analysis

Probabilistic Latent Semantic Analysis (PLSA) [32] is a well-known statistical topic model, which and whose variance algorithms [63, 60] are being widely used in practice. Let us analyze the complexity of *PET* by comparing it with PLSA.

For a collection of n documents that involves t topics and a vocabulary of w terms, it is easy to prove the complexity of PLSA is $O(ntwi)$ if we expect the EM procedure terminates after i iterations. Similarly, carrying out a Maximum A Posterior estimator, *PET* needs $O(nw)$ times computations for both of each E-step and M-step, if the cubic function (*i.e.*, Equation 4.6) is considered to be solved in constant time [64]. Based on the same assumption that the iterations end up after i rounds, *PET* takes $O(nwiT)$ time complexity for the T time points as a whole. By similar analysis, the time complexity of the extended model that tracks multi-events is $O(ntwiT)$.

Empirically, a popular event in social communities is only able to attract considerable public attention for a short period (*i.e.*, a small value of T), *e.g.*, the discussion of a news-related event on Twitter usually becomes trivial after the 30^{th} days after it was report for the first time, and the discussion of a movie-related event could be even shorter. Hence, the complexity of *PET* is reasonable and thus affordable in practice.

4.4 Experiments

We have introduced *PET*, a novel statistical model for Popular Event Tracking in social communities, and discussed its connections with two classical models [43, 62]. In this section, we show the effectiveness of our model with experiments on two different genres of data, *Twitter* and *DBLP*.

4.4.1 Popular Events Analysis on Twitter

Data Collection

Twitter[1] is a free social networking and micro-blogging service that enables its users to send and read messages known as *tweets*. Tweets are text-based posts of up to 140 characters displayed on the author's profile page and delivered to the author's *followers*. In this experiment, we create our testing data set by selecting $5,000$ users with follower-followee relationships and crawling down $738,826$ tweets displayed by these users during the period from Oct 2009 to Dec 2009. For

[1] http://www.twitter.com

each tweet, we extract its text contents and its followee (if available). Concretely, we consider each day as a time point; for each time point k, the document $d_{k,i}$ is obtained by concatenating tweets displayed by user i in day k; $g_k(i, j)$ equals to the number of tweets displayed by user i by following user j during the period from $k - 30$ to k.

Some simple statistics are presented as follows: (i) for each day, there are only average 37% users who display tweets; (ii) there are 12% days when less than 20% users display tweets; (iii) there are 58% tweets which have a followee; (iv) each user has average 10.2 followees. These statistics confirm our hypothesis stated in this paper: the information of an individual user sometimes is sparse, but individuals are strongly connected by networks.

Parameter, Baseline and Gold Standard

Parameter Setting. We implement our model as depicted in Section 4.2, with its three parameters set as $\lambda_T = 1$, $\lambda_A = 5$ and $\mu_E = 10$.

Baselines. We also implement four baselines for performance comparison:

1. ***HMM***. The first baseline is the state automation model based on HMM (Section 4.3.2), which is a variation of Jon Kleinberg's model [43]. Concretely, the observation x_k is the total frequency of event-related terms written by all users at time k, based on which the hidden state λ_k is estimated by Equation 4.10 resulting in the overall interest level after normalization. We believe this is a good representative of event tracking methods that do not consider the network effect.

2. ***Cont***. The second baseline is the contagion model [62] introduced in Section 4.3.2. Concretely, two users are neighbors in the contagion network if they have the follower-followee relationship. A user becomes newly infected if the number of infected users among her/his friends in last day is more than a pre-defined threshold. The recover ratio at day k is estimated according to the total frequency of event-related terms in D_k. This is a representative of network-based diffusion models that do not consider textual documents.

3. ***PET-1***. In Section 4.2, we explained three key observations that motivate the model design. To verify the effectiveness of Observation 1, we implement a special version of PET by removing the efforts of network structures, *i.e.*, we set $\lambda_A = 0$.

   PET-2. Similarly, to verify the effectiveness of Observation 2, we implement another special version of PET by eliminating the influence of history, *i.e.*, we keep $\lambda_T : \lambda_A = 1 : 5$, but set $\lambda_T = +\infty$.

Gold Standard. Events selected in this experiments can be generally categorized into movie-related events and news-related events. We utilize different sources as the gold standard for different categories:

1. ***Box Office***. For a movie related event, the box office is a trustworthy criterion to estimate the popularity. Hence, we extract the daily box office at Mojo [2] to be the gold standard for movie related events.

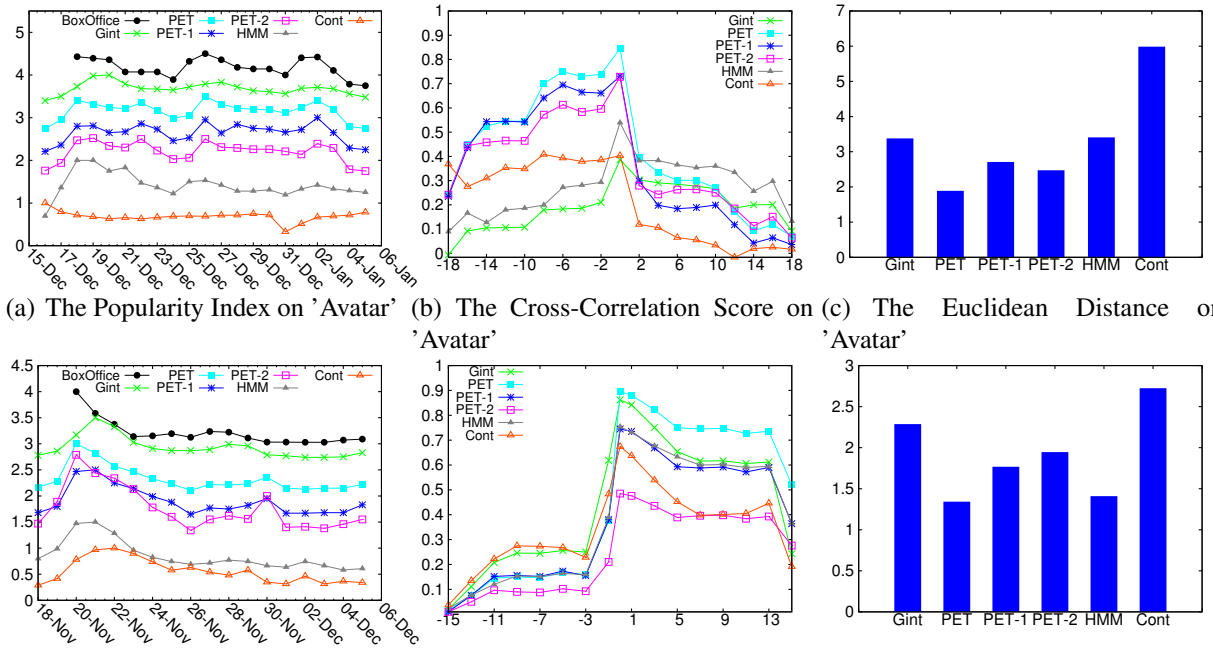
2. ***Gint***. For a news related event, the interest index by analyzing the search volume of Google can reflect the news' popularity well. Therefore, we use the interest index at Google Insight [3] to be the gold standard for news related events. Also, *Gint* is a baseline for movie related events.

Analysis on Popularity Trend

On this dataset, we track the overall popularity trend of events by using *PET*, *PET-1*, *PET-2*, *Cont*, *Box Office* (if available) and *GInt*, respectively. Four popular events are selected for analysis: two movie related events, *i.e.*, 'Avatar' and 'the Twilight Saga: New Moon', and two news related events, *i.e.*,'Tiger Woods Affair' and 'Copenhagen Climate Conference'. We selected these four events because their life cycle well overlaps with the time period of our *Twitter* data. Furthermore, we randomly select 5% users and the average popularity of selected users by each method is curved

[2] http://boxofficemojo.com/movies/
[3] http://www.google.com/insights/search/#

in Figure 4.1(a), 4.1(d), 4.2(a) and 4.2(d) for the four events, respectively. To make clearer comparisons, the curves in the same figure are normalized to the same scale and are shifted vertically by a distance. These modifications do not harm to our experiments, since the trend of each curve is completely reserved.



(a) The Popularity Index on 'Avatar' (b) The Cross-Correlation Score on 'Avatar' (c) The Euclidean Distance on 'Avatar'

(d) The Popularity Index on 'Twilight' (e) The Cross-Correlation Score on 'Twilight' (f) The Euclidean Distance on 'Twilight'

Figure 4.1: The Analysis of Popularity Trend: *PET* matches the gold standard best.

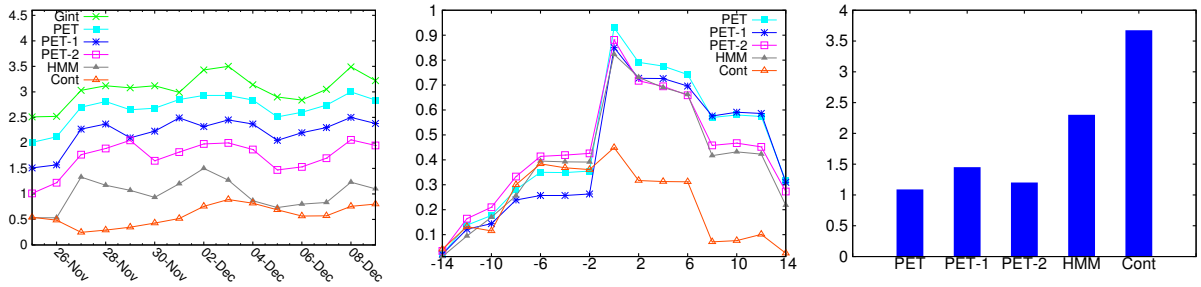
Evaluation Measures. We leverage cross-correlation score [13] and normalized Euclidean distance [28] to quantitatively measure the consistence of the trends to the gold standard. The two measures are defined as:

1. *Cross-Correlation Score*. The cross-covariance function between two time series $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$c_{\mathbf{xy}}(k) = \frac{\sum_{i=1}^{n-k}(x_i - \mu(\mathbf{x}))(y_i - \mu(\mathbf{y}))}{n}$$

for $k \in [0, n-1]$. and

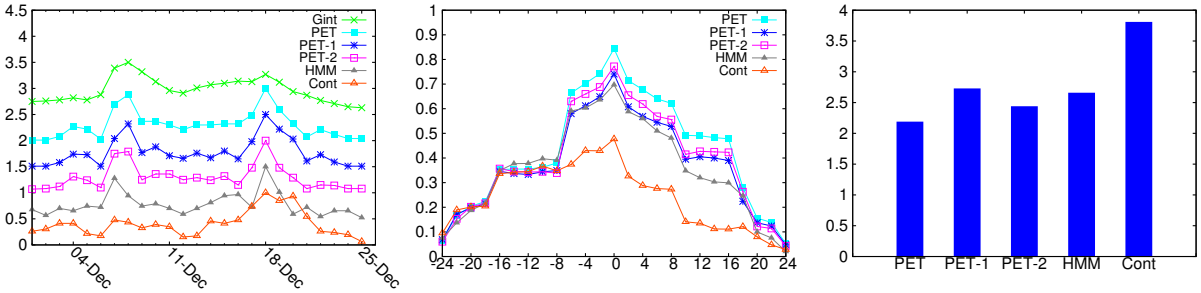
$$c_{\mathbf{xy}}(k) = \frac{\sum_{i=1-k}^{n}(x_i - \mu(\mathbf{x}))(y_i - \mu(\mathbf{y}))}{n}$$



(a) The Popularity Index on 'Tiger Woods'

(b) The Cross-Correlation Score on 'Tiger Woods'

(c) The Euclidean Distance on 'Tiger Woods'



(d) The Popularity Index on 'Copenhagen'

(e) The Cross-Correlation Score on 'Copenhagen'

(f) The Euclidean Distance on 'Copenhagen'

Figure 4.2: The Analysis of Popularity Trend (continue): *PET* matches the gold standard best.

for $k \in [-1, 1-n]$, where $\mu(\cdot)$ is the mean. The cross-correlation is the cross-covariance scaled by the variances of the two series:

$$r_{\mathbf{xy}}(k) = \frac{c_{\mathbf{xy}}(k)}{c_{\mathbf{xx}}(0) \cdot c_{\mathbf{yy}}(0)}$$

Figure 4.1(b), 4.1(e), 4.2(b) and 4.2(e) draw the cross-correlation curves between each method and the gold standard for the four events, respectively [4]. The higher the score is, the better the result is.

2. *Normalized Euclidean Distance*. A time series $\mathbf{x}$ is normalized by its mean $\mu(\mathbf{x})$ and variation $\sigma(\mathbf{x})$ as

$$x'(i) = \frac{x_i - \mu(\mathbf{x})}{\sigma(\mathbf{x})}$$

[4]We assume that the popularity of a movie at day k may be reflected at twitter at day $(k+1)$.

Then the Euclidean distance between two series **x** and **y** is defined as

$$d(\mathbf{x}, \mathbf{y}) = \mu(\mathbf{x'} - \mathbf{y'})$$

To the opposite of cross-correlation, a smaller distance means a better result.

Results Analysis. According to Figure 4.1 and 4.2, *PET* always has the highest cross-correlation score (see Figure 4.1(b), 4.1(e), 4.2(b) and 4.2(e)), and the smallest normalized Euclidean distance (see Figure 4.1(c), 4.1(f), 4.2(c) and 4.2(f)) with the gold standard. *PET* achieves the best performance for all events, since it evaluates the popularity evolution by comprehensively considering historic, textual and structured information. Missing any of the three components will inevitably decrease the accuracy.

1. *Cont* performs worst among all comparable methods, because it aims to answer the question in a different scenario: when can a local behavior spread to the whole network? As a contagion model, the behavior of one user can 'infect' another on the network via a long chain and by taking a long transfer time when local interaction is sufficient, so the popularity index evaluated by *Cont* at a certain time point could be the mixture of current user behaviors and the ones happened long time ago. However, such 'long chain' rule does not apply appropriately to popular events in online communities. For example, the popularity index of *Cont* at *Dec 28* in Figure 4.1(a) is unfavorably higher than the gold standard, because *Cont* mistakenly transferred some popularity from *Dec 26*. Also, *Cont* shows a smoother 'valley' at *Dec 5* in Figure 4.2(a) than the gold standard, because the 'valley' is neutralized by the 'peak' at *Nov 30*. Also, *Cont* ignores the connection between a user's interest level and the contents generated by her, which is contradict with our Observation 3.

2. *HMM* is less accuracy than *PET* at most time. There are at least two underlying reasons: (i) First, *HMM* is not able to detect coherent terms that are not given as event-related terms at the beginning, so it may underestimate the popularity due to missed coherent terms. For instance,

the popularity index of *HMM* at *Dec 28* in Figure 4.1(a) is much lower than the gold standard because people may mention the event 'Avatar' by using other words like 'James Cameron', 'film technology', 'box office', *etc*, rather than directly using the key words 'Avatar'. Also, such underestimation happened at *Dec 9* in Figure 4.2(d), since the query terms 'Copenhagen climate conference' is insufficient to describe the details of the conference. (ii) Second, similar to many other methods, *HMM* takes a sequence of aggregated counting data (*e.g.*, the total frequency of terms) as its observation. However, such aggregated data could not precisely stand for the popularity over the network. An intuitive example is: ten users claiming the same conclusion is definitely more trustworthy than one user repeating the conclusion by ten times. However, *HMM* treats the two situations indistinguishably.

Remind that the *PET* model is motivated by three key observations, *i.e.*, the interaction between users' interest and their connections, the influence to users' interests from their histories, and the interaction between user-generated contents and their interest levels. Now we go back to verify these intuitions:

1. To verify Observation 1, we implement *PET-1* by removing the network component from the *PET* model. Compared to *PET*, it shows two weakness due to the lack of network structures. On one hand, *PET-1* can not response sufficiently to sudden changes. When there is no textual information about a particular user at current time point, *PET-1* will set the new status of the user the same as the previous one, but *PET* can evaluate the new status more precisely by borrowing information from the user's neighbors. For example, the box office earnings increased a lot at *Jan 2* in Figure 4.1(a), but *PET-1* did not recognize such changes until *Jan 3*. On the other hand, *PET-1* is more fragile to reflect local noises. An intuitive example is: one tweet followed by ten users will be more influential than the same tweet without any follower. However, *PET-1* treats the two situations equivalently, so that the influence of 'isolated' tweets is unfavorably enlarged.

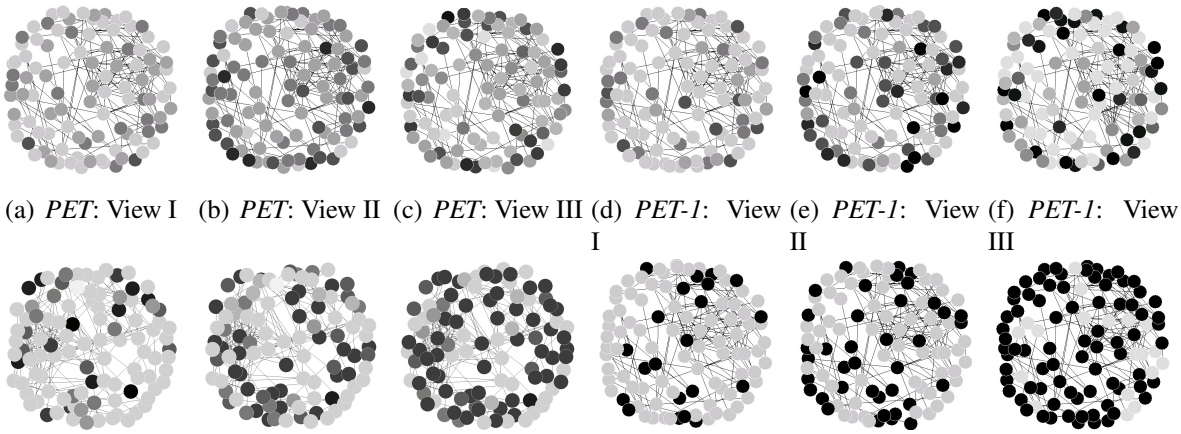
2. To verify Observation 2, another special version, *i.e.*, *PET-2*, is implemented by eliminating the historic efforts from the *PET* model. Without the historical reference, time-associated mistakes

may be amplified, resulting in false 'burstiness' or 'slump', *e.g.*, although all of *PET*, *PET-1* and *PET-2* have an unfavorable 'jump' at *Nov 30* in Figure 4.1(d), the one of *PET-2* is larger than the other two's, since *PET-2* has the least tolerance to the noises.

3. The network structure and document collection are the same to all events at a certain time point, but the popularity trend still appears variant due to different Dirichlet Prior in the topic model (Section 4.2.3). This is a good evidence to prove the connection between user-generated contents and their interest levels, *i.e.*, Observation 3.

Analysis on Network Diffusion

In this experiment, we study how events diffuse over networks. There is a burstiness from *Dec 16* to *Dec 18* in Figure 4.1(a) since *Dec 18* is the release date of 'Avatar' in north America. In Figure 4.3, we visualize the follow-followee networks on selected 100 users for *PET*, *PET-1* and *Cont* using the *NetDraw* software [5], where the color of a vertex represents the interest of a user and an edge stands for the follower-followee relationships. View I, II and III correspond to *Dec 16*, *17* and *18*, respectively [6].



(a) *PET*: View I (b) *PET*: View II (c) *PET*: View III (d) *PET-1*: View I (e) *PET-1*: View II (f) *PET-1*: View III

(g) *PET-2*: View I (h) *PET-2*: View II (i) *PET-2*: View III (j) *Cont*: View I (k) *Cont*: View II (l) *Cont*: View III

Figure 4.3: The Network Diffusion Analysis: *PET* generates the smoothest diffusion.

[5] http://www.analytictech.com/Netdraw/netdraw.htm

[6] View III uses a smaller scale of colors than View I and II, so as to avoid paleness of View I and II

As observed, *Cont* casts real-valued interest levels to binary ones, which inevitably causes the lost of accuracy, *e.g.*, compared to Figure 4.3(c), 4.3(f) and 4.3(i), Figure 4.3(l) seems to over-estimate the interest levels for selected users. Albeit the overall popularity evaluated by *PET*, *PET-1* and *PET-2* are similar as shown in Figure 4.1(a), vertices in networks of *PET* are smoothed via edges, which accords best with the situation in real world: *people's interests are inevitably influenced by their friends*.

Analysis on Content Evolution

Table 4.1-4.4 shows the topics extracted by *PET* that evolve along time. These results are interesting and reasonable.

Dec 14		**Dec 18**		**Dec 26**	
trailer	0.21	avatar	0.30	avatar	0.13
avatar	0.10	imax	0.06	imax	0.04
cameron	0.04	trailer	0.05	trailer	0.04
james	0.02	cameron	0.04	technology	0.03
sam	0.01	james	0.04	sam	0.02
director	0.01	alien	0.01	film	0.02
titanic	0.01	titanic	0.01	james	0.02

Table 4.1: The Content Evolution of 'avatar'

Nov 18		**Nov 20**		**Dec 22**	
moon	0.06	moon	0.17	moon	0.11
twilight	0.04	twilight	0.10	twilight	0.04
trailer	0.03	oprah	0.04	fantasy	0.02
chris	0.02	trailer	0.03	chris	0.02
stewart	0.01	vampire	0.03	saga	0.01
premiere	0.01	fantasy	0.02	women	0.01
taylar	0.01	midnight	0.01	milion	0.01

Table 4.2: The Content Evolution of 'twilight'

For example, in Table 4.1, users began to talk about 'Avatar' by introducing the movie's title, the actor 'Sam Worthington' and the director 'James Cameron' who was also the director of the movie 'Titanic'; in the release day of *Dec 18*, new terms appeared such as 'aliens' and 'iMax', and the term rank of 'trailer' dropped; when this movie became more and more famous, people

Nov 25		Nov 27		Dec 10	
rhapsody	0.07	tiger	0.11	tiger	0.10
muppets	0.06	woods	0.09	woods	0.06
bohemian	0.06	injure	0.04	brown	0.02
lambert	0.01	car	0.03	mistress	0.01
tiger	0.01	accident	0.03	golf	0.01
woods	0.01	championship	0.01	shame	0.01
playlist	0.01	hospital	0.01	divorce	0.01

Table 4.3: The Content Evolution of 'tiger woods'

Dec 07		Dec 15		Dec 18	
climate	0.02	oral	0.02	climate	0.04
copenhagen	0.01	council	0.02	copenhagen	0.03
conference	0.01	climate	0.01	conference	0.02
china	0.01	trade	0.01	reach	0.01
committee	0.01	copenhagen	0.01	summit	0.01
global	0.01	health	0.01	failure	0.01
warming	0.01	bill	0.01	agreement	0.01

Table 4.4: The Content Evolution of 'Copenhagen'

extended their discussion to the movie's historical significance, *i.e.*, its 3D 'film technology'.

Also, Table 4.3 shows the evolution of gossip on the golf star 'Tiger Woods': before *Nov 27*, the information about Tiger Woods was limited and inaccuracy; in *Nov 27*, the car accident was reported and people worried about his injury condition and golf competitions; after his affair was brought to light, people used words related to the 'scandal' such as 'brown' and 'mistress', blame his sexual abuse, and felt curious about the possible 'divorce'.

Again, by observing Table 4.4, we can easily find that people kept great attentions on the 'Copenhagen Climate Conference' when it was opened at *Dec 07*, but thought it was a 'failure' when the conference was closed at *Dec 18*.

In Table 4.2, the contents have not changed much for the movie 'twilight', which could be an evidence to explain why its box office earnings kept dropping since its release date.

4.4.2 Popular Events Analysis on DBLP

Data Collection. The Digital Bibliography and Library Project (DBLP) is a database which contains the basic bibliographic information of computer science publications[7]. In this experiment, we create our testing data set by selecting $12,949$ authors who published at least 10 papers in the conferences of database, data mining and information retrieval, and crawling down $500,417$ papers published by these authors during the period from the year of 1990 to 2008. Concretely, we consider one year as a time point, and titles and author lists are extracted to form the document collections and the co-author networks.

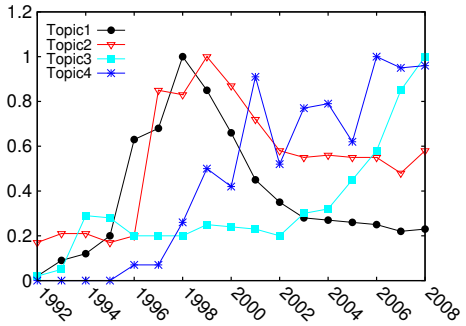


Figure 4.4: The Popularity Trend on DBLP

We select four research topics for case study, the top terms of which are: (i) 'frequent', 'itermset', 'pattern', 'mining', 'association', 'rule'; (ii) 'data', 'cube', 'OLAP', 'aggregation', 'dimensional', 'materialization; (iii) 'Web', 'mining', 'social', 'network', 'online', 'community'; and (iv) 'topic', 'languange', 'model', 'retrieval', 'probabilistic'.

Result Analysis. For each topic, the popularity trend is depicted by Figure 4.4, and co-author networks among top authors at two years are drawn by Figure 4.5. We find several interesting facts that refect the situations of the real world:

(i) Topic 1 was popular in last decade but was fading out recently, since frequent pattern (except graph pattern) mining techniques have been quite mature (Figure 4.4); 'Jiawei Han', 'Jian Pei', *etc*, form a stable 'cluster' that keeps doing research in this topic for many years (Figure 4.5(a)

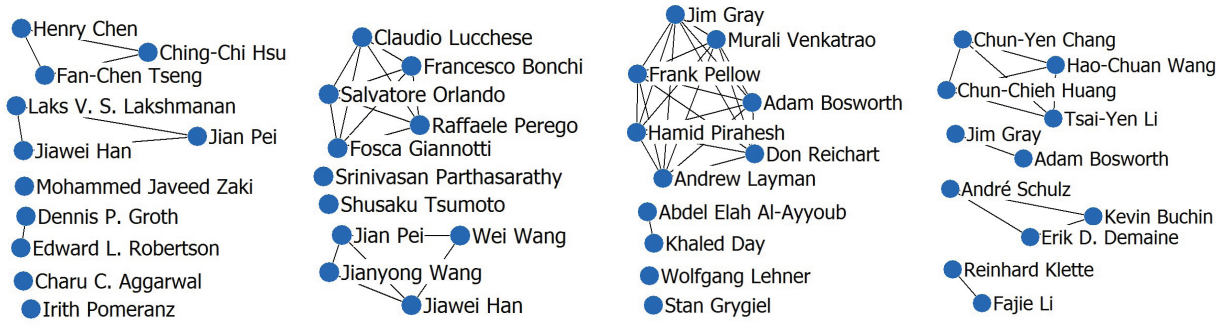
[7]http://www.informatik.uni-trier.de/∼ey/db

and 4.5(b)); ‘Raffaele Perego’, ‘Salvatore Orlando’, *etc*, create cooperation and become famous for their research on high performance data mining.

(ii) Topic 2 has a burstiness when ‘Jim Gray’ first introduced ‘data cube’ in 1996 (Figure 4.4), when most top-ranked authors in this topic coauthored with him (Figure 4.5(c)); and it attracted more attentions later (Figure 4.5(d)).

(iii) The popularity of topic 3 monotonically increases since web mining becomes more and more important (Figure 4.4); top-ranked authors even at two continuous years having little overlap (Figure 4.5(e) and Figure 4.5(f)) means it is a popular and thus competitive research direction.

(iv) Topic 4 has two rise-ups when ‘PLSA’ and ‘LDA’ was introduced in 1999 and 2002, respectively (Figure 4.4); some authors appear in both networks (Figure 4.5(g) and 4.5(h)), such as ‘ChengXiang Zhai’, ‘Jian-Yun Nie’, *etc*, but the co-authorship are less dense for the year of 2005 compared to the one of 2003.



(a) Topic 1 at Year 2001 (b) Topic 1 at Year 2006 (c) Topic 2 at Year 1997 (d) Topic 2 at Year 2007

(e) Topic 3 at Year 2007 (f) Topic 3 at Year 2008 (g) Topic 4 at Year 2003 (h) Topic 4 at Year 2005

Figure 4.5: The Evolution of Co-author Networks on DBLP

Summary of Experiments To sum up, we evaluate the performance of *PET* on two real datasets *Twitter* and *DBLP*, and compare it with four baseline methods. By comprehensively considering historic, textual and structured information into a unified model, *PET* generates the most accurate trends, the smoothest diffusion, and meaningful content evolution for popular events in social communities.

4.5 Conclusion

In this chapter, I propose the novel problem of popular events tracking in a social community. Given a stream of network structures, an associated stream of text documents, and the primitive form of events, we could track the popularity of the events on the network and content revolution of the events over time. We make several key observations about how the interest, topics and network structures mutually influence each other, and propose a novel statistical model that can handle all the constraints. The proposed model, PET, not only provides a unified probabilistic framework to model different factors in modeling the evolution of interests and contents, but also covers classical models as special cases. Comprehensive experimental studies on two real-world datasets show that our approach outperforms existing ones, and two of them are actually special cases of our model in certain circumstances. Our approach can potentially enable many more informative analysis of certain topics on specific networks, and interesting real-world applications.

Chapter 5

Diffusion and Evolution of Popular Events in Social Communities

In this section, I propose a novel statistical model for topic-based information diffusion and evolution (TIDE). Specifically, a mixture model is introduced to model the generation of text according to the diffusion and the evolution of the topic, while the whole diffusion process is regularized with user-level social influences through a Gaussian Markov Random Field. The discovery of novel aspects and the diffusion paths of the topic can be done by the joint inference of topic diffusion and evolution in TIDE.

When a topic is introduced into the community by a user, other users read the documents she wrote (*e.g.*, tweets, blogs, scientific papers, *etc*) and adopt the topic by writing about it in their own articles. They may or may not cite the original document, or they may cite it together with other documents. Although topics are spread among documents instead of through social connections, we consider it is much more likely for users to adopt ideas from their social connections (*e.g.*, friends, people they follow, or people they have cited before) than from a stranger. Each document can not only adopt content from documents that influenced it, but also include novel perspectives into the topic, and pass on the *'innovation'* to other documents. The meaning of the topic is thus evolving over time. The goal of the joint inference of topic diffusion and topic evolution is to identify the *'real'* paths through which the topic propagates, and also identify the time specific versions of the topic.

Organization. The rest of this section is organized as follows: Section 5.1 formally defines the problem of TIDE, as the solution of which a statistical model is proposed in Section 5.2. We present experiments and results in Section 5.3.

5.1 Problem Formulation

In this section, we formally define the task of inferring the diffusion process and tracking the evolution of topics in social communities. We begin with a few key concepts as follows.

Definition 5.1.1. **Social Network**. A *network* is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of vertices and $\mathcal{E}$ is a set of edges among V. Particularly in a *social network*, a vertex corresponds to a user, and an edge $e = (i, j)$ stands for a connection (or a tie) between two users i and j. The strength of the tie (i, j) is defined as a non-negative value $g(i, j)$. An edge can be either *directed* or *undirected*.

Definition 5.1.2. **Document Collection**. A textual document d_i in a *document collection* $\mathcal{D} = \{d_i\}_{i=1}^{M}$ is defined as a bag of words from a fixed vocabulary $\mathcal{W} = \{w_k\}_{k=1}^{L}$. That is, $d_i = \{c(d_i, w_k)\}_{k=1}^{L}$, where $c(d, w)$ denotes the number of occurrences of word w in d.

Definition 5.1.3. **Social Community**. A *social community* is defined as the union of a social network $\mathcal{G}$ and a user-generated document collection $\mathcal{D}$, saying $\{\mathcal{G}, \mathcal{D}\}$. Each document $d_i \in \mathcal{D}$ is associated with an *author* a_i in $\mathcal{G}$ and a *time-stamp*s $t_i \in 1..T$.

Definition 5.1.4. **Topic**. A semantic *topic* θ observed in a particular time period is defined as a multinomial distribution of words $\{p(w|\theta)\}_{w \in \mathcal{W}}$ with the constraint $\sum_{w \in \mathcal{W}} p(w|\theta) = 1$.

Definition 5.1.5. **Theme**. We define a general and coherent *theme* discussed in a social community as a stream of time-stamped topics $\Theta = \{\theta_t\}_{t=0}^{T}$. We call θ_0 the *primitive topic*, which represents the original content of the theme prior to the discussions of the social community. $\theta_{t>0}$ are time variant versions of θ_0, which are gradually developed in the discussions of the social community. That is, θ_t is the snapshot of Θ at time t, which represents the novel aspect of the theme appearing at time t. Altogether Θ represents the origin and evolution of the contents of the theme over time.

While the text content of individual document can be explicitly observed, the general semantics of the time-variant topics and the adoption of the topic(s) in a document is implicit. What else remains implicit is the source adopted in a document. There could naturally be multiple sources: a

document can be influenced by a few other documents, thus inherit the topic from those documents. The influence of some sources can be more salient than the influence of others. A document could also introduce original perspectives of the topic without being influenced by any existing document. The existence and strengths of the influence among documents assemble the actual diffusion process of the topic, which is formally defined as a *diffusion graph*.

Definition 5.1.6. **Diffusion Graph**. Given a theme Θ, we define a *diffusion flow* from one document d_j to another d_i ($t_j < t_i$) as the likelihood that d_i adopted the topic of Θ due to the influence of d_j. The strength of such a diffusion flow is denoted as a positive value $\pi_{i,j}$. Note that d_i can also introduces its novel perspective to Θ. In this case, we assume there is a diffusion flow into d_i from the time-stamped topic θ_{t_i}, with a strength $\pi_{i,\theta}$. Therefore, we define *diffusion vector* π_i as a vector of the strength of all the diffusion flows into d_i, *i.e.*, $\pi(i) = \{\pi_{i,j}\}_{d_j \in \mathcal{D}} \cup \{\pi_{i,\theta}\}$, with the constraint $\sum_{d_j \in \mathcal{D}} \pi_{i,j} + \pi_{i,\theta} = 1$. The union of diffusion flows into all documents in $\mathcal{D}$ assembles the *diffusion graph*, *i.e.*, $\Pi = \{\pi(i)\}_{d_i \in \mathcal{D}}$. Clearly, the graph Π is both weighted and directed.

Although the actual diffusion graph is unobserved, there are proxy networks that convey weaker signals in social communities. In many scenarios, a reference network (denoted as $\mathcal{R}$) of the documents can be observed, for example the citation network of scientific publications, the hyperlink network of blog articles, or the tweet network posted by follower-followees. Intuitively, the diffusion network should correlate well with such a reference network because a document is very likely to be influenced by documents it cites. However, the actual diffusion network could still be substantially different from $\mathcal{R}$, because many real influential references are covered up, and many explicitly cited ones do not represent the true influence. Another signal is the social network structure. An author is likely to follow the work of his social connections, such likely to adopt topics and ideas from the documents they generate [60, 79]. We call the set of documents pointing to d_i in the reference network as d_i's *reference set*, denoted as $r(i) \subset \mathcal{D}$. When no signal of citation or social communication is available, r_i can be simply instantiated as all documents with a

time-stamp prior to t_i. When such a reference network is available, we assume $\pi_{i,j} = 0$ if $j \notin r(i)$. Clearly, we also have $\pi_{i,i} = 0$.

Based on the definitions of concepts above, we can formalize the two major tasks of tracking **the diffusion and evolution of topics in social communities**. Given the input of a social community $\mathcal{G}$, a user-generated document collection $\mathcal{D}$, and the primitive topic θ_0 defining a theme, we aim to:

Task 1: Infer the Diffusion Graph. In this task, the goal is to discover the latent diffusion flow graph documents (and topics) (*i.e.* Π). The result of this task can be used to answer (i) *the source(s) of topic in a document*: to what extent the document is influenced by other documents, and (ii) *the degree of originality in a document*: how much novel perspectives the document introduces to the topic.

Task 2: Track Topic Evolution. In this task, the goal is to infer the time-variant versions of topics (*i.e.*, $\{\theta_t\}_{t=1}^{T}$) of a theme. By inferring Θ given θ_0, we expect to keep track of the new developments of the theme, understand its evolution over time, and better understand how it influences documents, *etc.*

The two tasks are challenging in many ways. First, although recently there are extensive studies on inferring the social influence on explicit behaviors at the user level [17, 83, 1, 26, 89], there is limited progress in the analysis of the influence on the adoption of latent topics at document level [47, 85]. Even though topic diffusion occurs at document level, the influence along the social network structure is playing an non-negligible role. There is however little existing wisdom on how to bridge document networks with social networks. The implicit nature of topics have made the problem even harder. Second, the inference of topic diffusion cannot be done independently to the tracking of topic evolution. Along with the diffusion process of the topic, new contents are introduced into the topic, making the semantics of the topic evolving over time. Without understanding the shift of topic contents, it is impossible to accurately detect the adoption of the topic in documents especially after a substantially long time. Moreover, since usually there are

limited labeled examples, the solution model should be unsupervised. All of these challenges require us to propose a unified model that takes social connections, textual contents, influence among documents, and temporal information into consideration. In the following section, we propose such an integrative model and present the joint inference of topic diffusion and evolution.

5.2 Proposed Models

In this section, we propose a novel and integrative probabilistic model of Text-based Information Diffusion and Evolution (TIDE) in social communities. Based on TIDE, we present the joint inference of the diffusion graph and the evolution of arbitrary topics.

5.2.1 Intuitions and the General Model

The general model of TIDE is designed based on a few key observations in social communities.

Observation 4. *Diffusion and Contents*. When there is a significant diffusive flow between two documents, or there is a significant influence on one document on the other, the content of these two documents tend to be highly related. On the other hand, if two documents talk about different subjects, there is unlikely salient influence or significant diffusion flow between them even if one cites the other [75]. *W.l.o.g.*, we can assume that the content of a document depends on the documents which have influenced it.

Observation 5. *Diffusion and Social Connections*. Information transmission is a complex social-psychological behavior [58], *e.g.*, there exist persistent interests of users [68]. The diffusion process among documents is likely to be regularized by social connections of their authors. Indeed, an author is more likely to follow the work of her friends and thus adopt topics and ideas from a friend instead of from a random author. The diffusion flows among documents are thus dependent to the social network of authors.

Observation 6. *Diffusion and Evolution*. As the diffusion proceeds, both the semantics of the topic and the regularization effect of the social network of users evolve over time. If an aspect in a document never appear in any of its potential references (either papers it cites or all existing papers exposed to its author), it is likely to be original ideas introduced by the document, which contributes to the evolution of the general theme. Meanwhile, the strength of influence through old social connections would decay after a reasonably long time.

Given a collection of authored and time-stamped documents $\mathcal{D}$, a social community $\mathcal{G}$ of users who published these documents, and a primitive topic θ_0 representing the original semantics of a theme, we aim at inferring the latent stream of topics Θ and the diffusion graph Π. Based on our observations above, the task of TIDE is then cast as the joint inference of the posterior of Θ and Π:

Formally, our object becomes to infer:

$$P(\Pi, \Theta|\mathcal{G}, \mathcal{D}, \theta_0) \propto P(\Theta|\Pi, \mathcal{D}, \theta_0) \cdot P(\Pi|\mathcal{G}) \tag{5.1}$$

Based on our observations, here we assume that the generation of the diffusion graph (only) depends on the social network structure, while the evolution of topics depends on the documents, the diffusion process, and of course the original version of the topic. We denote the first component of Equation 5.1 as the *topic model* and the second as the *diffusion model*. Please note that although TIDE can be easily extended to model the mixture of multiple topics (similar to LDA [9]), we only present the primitive case to model only one given topic. Our focus is to model the diffusion and evolution of any given topic instead of the discovery of multiple topics. We leave the modeling of multiple topics in our future work.

In the topic model, a *mixture model* is designed to extract the topic snapshots (time-variant versions) of the theme (Section 5.2.2). In the diffusion model, we introduce a *Gaussian Markov random field* based on *graph projection* to model the dependency of diffusion flows on social connections (Section 5.2.3). Finally, the inference of the combined model is discussed in Section 5.2.4.

5.2.2 The Topic Model

It is difficult to directly compute the posterior of topics Θ. We make the following transformation such that

$$P(\Theta|\Pi, \mathcal{D}, \theta_0) \propto P(\mathcal{D}|\Theta, \Pi, \theta_0) \cdot P(\Theta|\theta_0), \tag{5.2}$$

where the introduction of new aspects to the topic (*i.e.*, the time-variant topic snapshots) does not depend on the diffusion flows.

We consider a typical generative process of $\mathcal{D}$: each document d_i is generated from a mixture model. When writing each word in d_i, one first chooses a component model from the mixture with a certain probability; once the component model θ is selected, a word is sampled according to the word distribution of θ.

We first introduce a background component model θ_B estimated from the entire collection that explains the generation of common English words in the document d_i. The rest component models are designed based on the diffusion flows. Specifically, we introduce a component model for each document d_j that could have potentially influenced d_i. There is a non-trivial diffusion flow from d_j to d_i, and d_i could inherit the topic of d_j according to the strength of this diffusion. These component models can be estimated simply using a maximum likelihood estimator on the corresponding d_j. Finally, we introduce a component model to explain the novel aspects introduced by the document d_i, *i.e.*, the aspect that is not influenced by any existing document. We assume that this aspect is generated directly from the latent topic at the time that d_i is written (θ_{t_i}). In other words, the original content is diffused from the topic directly to the document instead of from other documents. We assume that the probability of choosing each component is proportional to the strength of the diffusion vector, *i.e.*, $\pi(i)$.

Formally, the probability of generating a word w in d_i is:

$$p(w|d_i) = (1 - \lambda_B)(\sum_{j \in r(i)} \pi_{i,j} p(w|\theta_{d_j}) + \pi_{i,\theta} p(w|\theta_{t_i})) + \lambda_B p(w|\theta_B)$$

where λ_B is a predefined parameter that fixes the sampling probability of the background model. Note that for documents $d_j \notin r(i)$, we have $\pi_{i,j} = 0$. The likelihood of the collection $\mathcal{D}$ is given as:

$$P(\mathcal{D}|\Pi, \Theta, \theta_0) = \prod_{d_i \in \mathcal{D}} \prod_{w \in \mathcal{W}} p(w|d_i)^{c(w,d_i)}$$

We then consider the generation of the time-variant versions of the topic, Θ. In TIDE, the primitive topic θ_0 is realized as a conjugate Dirichlet prior of the time-variant topic model θ_t: $Dir(\{1 + \mu_E p(w|\theta_0)\}_{w \in \mathcal{W}})$. By doing so, we regularize these time-variant topic snapshots so that they can reflect the novel aspects of the theme, but do not shift away from it. μ_E indicates how much we rely on the prior. Formally,

$$P(\Theta|\Pi) = \prod_{t \in 1..T} p(\theta_t|\theta_0) = \prod_{t \in 1..T} \prod_{w \in \mathcal{W}} p(w|\theta_t)^{\mu_E p(w|\theta_0)}$$

5.2.3 The Diffusion Model

Comparing to the modeling of topic evolution, the modeling of diffusion graph ($P(\Pi|\mathcal{G})$) is less straightforward. Intuitively, the diffusion graph Π should be regularized by the social network $\mathcal{G}$, as social influence plays an important role in topic diffusion. However, Π is a network of *documents* while $\mathcal{G}$ is a network of users. This makes it hard to model the regulation effect of $\mathcal{G}$ on Π. We need a bridge between the two heterogeneous networks, for which we introduce the operation of *graph projection*.

Definition 5.2.1. **Graph Projection**. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two graphs, a projection $f : \mathcal{G}_1|\mathcal{G}_2 \to \mathcal{G}_1'$ is called a *graph projection* if:

1. $\mathcal{V}(\mathcal{G}_1') = \mathcal{V}(\mathcal{G}_2)$.

2. $\forall v \in \mathcal{V}(\mathcal{G}_1'), \exists u \in \mathcal{V}(\mathcal{G}_1)\ s.t.\ v \in f(u)$.

3. $\forall e = (u, v) \in \mathcal{E}(\mathcal{G}_1'), \forall u' \in f(u)\ and\ v' \in f(v), e' = (u', v') \in \mathcal{E}(\mathcal{G}_1')$.

Through graph projection, two networks are endowed with the same vertex set, so that the comparison of them becomes more succinct and natural. Note that there are two asymmetric projection directions: 1) projecting $\mathcal{G}$ into a document network and using it as *a priori* of Π, or 2) projecting Π into a social network and consider the generation of such a social network based on $\mathcal{G}$. Since the document collection $\mathcal{D}$ is commonly much larger than the set of user $\mathcal{V}(\mathcal{G})$, projecting the document network into a social network is at inevitable risk of losing information. Although this doesn't rule out the second direction of graph projection, in this work we consider the first direction: the projection of $\mathcal{G}$ into a document network.

Let's denote Π' as the document network projected from $\mathcal{G}$, s.t.

$$P(\Pi|\mathcal{G}) = P(\Pi|\Pi') = P(\{\pi(i)\}_{d_i \in \mathcal{D}}|\Pi').$$

The remaining issue is how to fold the $\mathcal{G}$ into Π' and how to model the generation of Π based on Π'. Note that like Π, we can also denote $\Pi' = \pi'(i)_{d_i \in \mathcal{D}}$. We start with the generative model $P(\Pi|\Pi')$.

Gaussian Graphical Models (GGM) [90] are classical models used to explain the generation of networks, which could be an ideal solution of our problem. In a typical GGM, each nodes in the graph is modeled as a random variable, for example a vector of k features. In our scenario, such a vector can be implemented as the diffusion vector $\pi(i)$. The joint distribution of all these variables (in our case, $P(\pi(i))$) is assumed to be a multivariate Gaussian. Each edge in Π' stands for the conditional dependency between two Gaussian variables, thus the graph structure Π' corresponds to the inverse covariance matrix.

However, the computational complexity of such a graphical model usually scales cubically with the number of variables, and therefore becomes intolerant even for a moderate size of dataset. To make our model practical, we introduce an independence assumption: the diffusion vector of one document is independent to the others. By doing so, we can simplify the generative model of

Π as

$$P(\Pi|\Pi') = \prod_{d_i \in \mathcal{D}} P(\pi(i)|\pi'(i)) \tag{5.3}$$

Here $\pi'(i) = \{\pi'_{i,j}\}_{j \in r(i)} \cup \{\pi'_{i,\theta}\}$ is a conjugate prior vector, indicating the expected value of $\pi(i)$. Since Π' is projected from $\mathcal{G}$, $\pi'_{i,j}$ represents the social influence between a_j (the author of d_j) to a_i, which decays over time. By doing this, the document-level influence is regulated by the social tie at the user level.

Formally, we define $\pi'_{i,j} = \frac{1}{Z(\pi'(i))} g(a_i, a_j) \cdot e^{-\frac{t_i - t_j}{\alpha}}$ by consolidating an exponential time model with $\mathcal{G}$ [1]. Intuitively, documents with higher authority is likely to introduce more original content. We thus define $\pi'_{i,\theta} = \frac{1}{Z(\pi'(i))} Aut(a_i)$, where $Aut(a_i)$ is an estimation of the authority of d_i. $Z(\pi'(i))$ is a normalization factor such that $\sum_{d_j \in \mathcal{D}} \pi'_{i,j} + \pi'_{i,\theta} = 1$.

Given the design of $\pi'(i)$, the computation of $P(\pi(i)|\pi'(i))$ is still non-trivial because of the dependency between the dimensions of $\pi(i)$. We introduce a *Gaussian Markov Random Field* [70] to model the conditional probability $P(\pi(i)|\pi'(i))$ for each d_i.

Definition 5.2.2. **Gaussian Markov Random Field (GMRF)** . A random vector $\S = (x_1, x_2, \cdots, x_n)^T$ is called a GMRF *w.r.t.*the graph $\mathcal{G} = (\mathcal{V} = \{1, 2, \cdots, n\}, \mathcal{E})$ with the mean μ and the precision matrix $\mathcal{Q}_\S$, iff the density of $\S$ has the form

$$P(\S) = (2\pi)^{-n/2} |\mathcal{Q}_\S|^{1/2} e^{-\frac{1}{2}(\S - \mu)^T \mathcal{Q}_\S (\S - \mu)}$$

and $\mathcal{Q}_\S(i, j) \neq 0 \Leftrightarrow (i, j) \in \mathcal{E}$ for all $i \neq j$.

In our case, the random vector is the diffusion vector $\pi(i)$, with the mean as the prior vector $\pi'(i)$. The precision matrix $\mathcal{Q}_{\pi(i)}$ corresponds to the similarities between the dimensions of $\pi(i)$ (documents and topic snapshots), which can be realized as the content similarities of corresponding

[1] Other decay functions are also applicable [20].

θ_{d_j}'s and θ_t's. Computationally, $P(\pi(i)|\pi'(i))$ is defined as:

$$P(\pi(i)|\pi'(i)) \propto e^{-\frac{1}{2}\sum_{i',j' \in \{r(i)\}\cup\{\theta\}} (\pi_{i,i'}-\mu_{i,i'})\mathcal{Q}_{\pi(i)}(i',j')(\pi_{i,j'}-\mu_{i,j'})}$$

5.2.4 Parameter Estimation

Given our model defined above, we can fit the model to the data and estimate the parameters using a Maximum A Posterior estimator [77]. Expectation Maximization (EM) algorithm [59] is applied, which iteratively computes a local maximum of the posterior. Computationally, the log likelihood we want to maximize is:

$$\begin{aligned}
& E_{\Lambda^{(n-1)}}\{\log p(C|\Lambda)p(\Lambda)\} \qquad (5.4)\\
\propto & \sum_{d_i,w,d_j\in r(i)} c(d_i,w)(1-z^{(n)}_{d_i,w}(\theta_B))z^{(n)}_{d_i,w}(\theta_{d_j})\log((1-\lambda_B)\pi_{i,j}p(w|\theta_{d_j}))\\
& +\sum_{d_i,w} c(d_i,w)(1-z^{(n)}_{d_i,w}(\theta_B))z^{(n)}_{d_i,w}(\theta_{t_i})\log((1-\lambda_B)\pi_{i,E}p(w|\theta_{t_i}))\\
& +\sum_{d_i,w} c(d_i,w)z^{(n)}_{d_i,w}(\theta_B)\log(\lambda_B p(w|\theta_B)) + \mu_E\sum_{\theta_t,w} p(w|\theta_0)\log p(w|\theta_t)\\
& -\frac{\mu_G}{2}\sum_{d_i}\sum_{i',j'\in\mathcal{N}(i)} (\pi_{i,i'}-\mu_{i,i'})\mathcal{Q}_{pi(i)}(i',j')(\pi_{i,j'}-\mu_{i,j'})
\end{aligned}$$

Here μ_G is a weight combining two components, and we use terms $z_{d_i,w}(\cdot)$ instead of $p(z_{d_i,w}=\cdot)$ for better equation display.

In the E-Step, we compute the expectation of the hidden variables:

$$z^{(n)}_{d_i,w}(\theta_{d_j}) = \frac{\pi^{(n-1)}_{i,j}p(w|\theta_{d_j})}{\sum_{j'\in r(i)}\pi^{(n-1)}_{i,j'}p(w|\theta_{d_{j'}}) + \pi^{(n-1)}_{i,\theta}p(w|\theta_{t_i})}$$

$$z^{(n)}_{d_i,w}(\theta_{t_i}) = \frac{\pi^{(n-1)}_{i,\theta}p^{(n-1)}(w|\theta_{t_i})}{\sum_{j'\in r(i)}\pi^{(n-1)}_{i,j'}p(w|\theta_{d_{j'}}) + \pi^{(n-1)}_{i,\theta}p(w|\theta_{t_i})}$$

$$z^{(n)}_{d_i,w}(\theta_B) = \frac{\lambda_B p(w|\theta_B)}{(1-\lambda_B)(\sum_{j'\in r(i)}\pi^{(n-1)}_{i,j'}p(w|\theta_{d_{j'}}) + \pi^{(n-1)}_{i,\theta}p(w|\theta_{t_i})) + \lambda_B p(w|\theta_B)}$$

In the M-step, given the expectation of the hidden variables, we get the best parameters $p(w|\theta_t)$ as:

$$p(w|\theta_t) = \frac{\sum\limits_{d_i,t_i=t} c(d_i,w))(1-z^{(n)}_{d_i,w}(\theta_B))z^{(n)}_{d_i,w}(\theta_t) + \mu_E p(w|\theta_0)}{\sum\limits_{w'}\sum\limits_{d_i,t_i=t} c(d_i,w'))(1-z^{(n)}_{d_i,w}(\theta_B))z^{(n)}_{d_i,w'}(\theta_t) + \mu_E p(w'|\theta_0)}$$

By integrating Lagrange multipliers [59] f_i for each $d_i \in \mathcal{D}$, the inference of $\pi(i)$ boils down to solve a group of cubic equations:

$$\pi^2_{i,*} + \beta_{i,*}\pi_{i,*} + \gamma_{i,*} = 0, \quad * \in r(i) \cup \{\theta\} \tag{5.5}$$

where

$$\begin{aligned}
\beta_{i,*} &= \frac{\sum\limits_{*' \neq *} (\mathcal{Q}_{\pi(i)}(*,*') + \mathcal{Q}_{\pi(i)}(*',*))(\pi^{(n-1)}_{i,*'} - \mu_{i,*'})}{2Q_{\pi(i)}(*,*)} - \mu_{i,*} + \frac{f_i}{\mu_G \mathcal{Q}(i)_{*,*}} \\
\gamma_{i,*} &= -\frac{\sum\limits_{w} c(d_i,w)(1-z^{(n)}_{d_i,w}(\theta_B))z^{(n)}_{d_i,w}(\theta_{d_j})}{\mu_G \mathcal{Q}(i)_{*,*}}
\end{aligned}$$

Let $\pi_{i,*} = \frac{-\beta_{i,*}+\sqrt{\beta^2_{i,*}-4\gamma_{i,*}}}{2}$ be the root of Equation 5.5. It is easy to prove that $\pi_{i,*}$ can be arbitrarily close to zero when $f_i \to +\infty$ and arbitrarily large when $f_i \to -\infty$. Also, the derivative $\frac{\partial \pi_{i,*}}{\partial f_i} = \frac{1}{2}\left(-1 + \frac{\beta_{i,*}}{\sqrt{\beta^2_{i,*}-4\gamma_{i,*}}}\right) < 0$. Hence, it is guaranteed that there exist valid solutions for the group of equations that satisfy the constraint $\sum_{*\in r(i)\cup\{\theta\}} \pi_{i,*} = 1$ for each d_i in $\mathcal{D}$.

5.3 Experiments

In this section, we evaluate the effectiveness of our TIDE model on synthetic datasets as well as data collected from two real-world social communities, *i.e.*, *DBLP* [2] [84] and *Twitter* [56].

[2] http://.org/DBLP_Citation

5.3.1 Experimental Setup

Data Collections

The DBLP Dataset ([84]). The Digital Bibliography and Library Project (DBLP) is a web accessible database of the bibliographic information of computer science publications. In this experiment, we use a collection of DBLP articles augmented with citation information, released by the Arnetminer group, which contains $1,632,442$ publications by $1,741,170$ researchers with $2,327,450$ citations. After filtering out papers without text or citation information, $243,425$ papers and $246,839$ authors are retained. This dataset represents a typical academic community, with a social network of authors (with coauthoring and citation relations) and a collection of scientific papers.

The Twitter Dataset ([56]). Twitter is a well known social networking and micro-blogging community. In this experiment, the Twitter dataset was crawled down by the DAIS group at University of Illinois , which contains $5,000$ socially connected users and their most recent 200 tweets posted before Nov. 23, 2010. Totally, there are $103,968$ one-way following relations, and $51,032$ pairs of friends (mutual following relations). This dataset represents a typical social community with a directed social network (defined by following relations) and a collection of tweets.

Synthetic Dataset. The lack of ground truth on real world dataset makes it hard to evaluate the model performance quantitatively. To achieve quantitative evaluation, we construct a synthetic dataset which simulates the diffusion of $1,000$ themes. For each theme, we extract a subgraph of $1,000$ authors from the *DBLP* dataset using breath first search from a random seed author. This subgraph is used to simulate the social network in which the theme diffuses. We then randomly attach $1,859$ empty and time-stamped documents to the authors in this network [3]. We then simulate a diffusion graph of the $1,859$ documents that is regularized by the simulated social network structure. Specially, we first randomly generate a network of the $1,859$ documents using Erdos/Renyi model, with the average degree of 5 (consistent with the real statistics in the DBLP dataset). The

[3] According to the statistics on our *DBLP* dateset, each researcher has 1.859 first-authored publications in average.

direction of each edge is determined by the time stamps of the documents (always points to a "newer" document). We then weight each edge based on the social connections of the authors of the two document plus a random effect. This directed and reweighed random network simulates the real diffusion network among documents. For each theme, we also simulate a sequence of 10 evolving topic snapshots based on the dynamic topic models [8]. Finally, the text content of each document is generated by a simple mixture model with all documents that have "influenced" this document as well as the corresponding topic snapshot.

Baselines

The NetInf Model [26]. NetInf is a typical model that infers the diffusion network of explicit user behaviors. Given the time stamps at which individuals adopt a behavior, *NetInf* identifies the optimal general network of users that best explains the observed adoptions. Comparing to TIDE, *NetInf* is trying to infer the general social network structure according to the observation of the propagations of a group of events, while *TIDE* infers the theme-specific diffusion graph with the help of a general social network. Note that *NetInf* doesn't consider text information, thus cannot track topic evolution.

If we treat each term with a positive probability in the primitive topic as an explicit event/behavior, then a document adopts that behavior explicitly if the term appears in the document. We are then able to infer the optimal document network using *NetInf*. This optimal network is easily converted into a diffusion graph by endowing each edge with equal flow volume.

The IndCas Model [71]. The second baseline is a deviation of the independent cascade model stated in [71], where the probability for an active document to infect another is proportional to the strength of the social connection between their authors with an exponential decay effect [20] (see Section 5.2.3). We convert these probabilities into a diffusion graph where the diffusion flow from d_j to d_i is proportional to the probability that d_j infects d_i.

The TIDE- Model. To evaluate the effectiveness of social connections in our models, we imple-

ment a special version of *TIDE* by removing the regularization term with the network structure, *i.e.*, by setting $\mu_G = 0$.

We believe *NetInf* and *IndCas* are good representatives of diffusion inference models of explicit behaviors, which do not consider textual information or the evolution of topics. *TIDE*- on the other hand ignores the effects of social connections.

5.3.2 Experiments on Synthetic Data

The goal of the experiments on synthetic data is to quantitatively evaluate how well each method can (i) infer diffusion graphs, (ii) estimate contribution of novelty (if possible), and (iii) discover snapshots in topic evolution (if possible). Given the simulated social community (the social network, the document collection, and the primitive topic), our goal is to recover the diffusion graph and the topic snapshots. The parameters in the *TIDE* model are set empirically as $\mu_E = 10$, $\alpha = 30$, and $\mu_G = 10$.

Analysis on Information Diffusion

We first look at how successful models are at inferring diffusion graphs. Let us first introduce the evaluation metrics.

Definition 5.3.1. **Graph KL-Divergence**. The symmetrized Kullback-Leibler divergence [45] is a classic measure of the difference between two probability distributions. We extend the SKLD and define an evaluation metric to measure the discrepancy between two diffusion graphs $\Pi_{\mathcal{P}}$ and $\Pi_{\mathcal{Q}}$ on the same document collection $\mathcal{D}$:

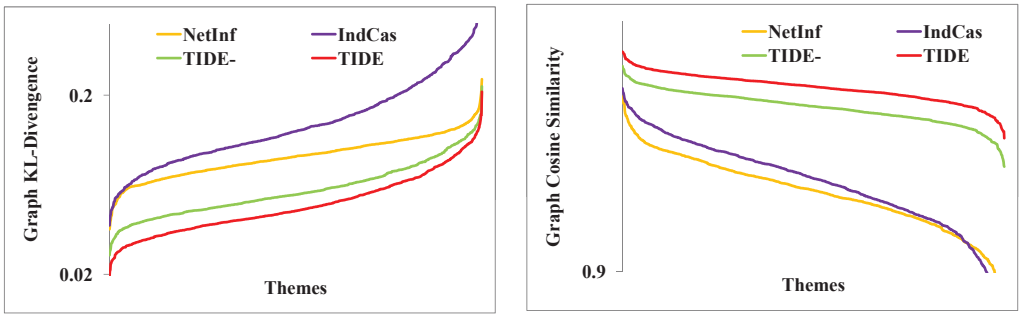
$$GD_{KL}(\Pi_{\mathcal{P}}, \Pi_{\mathcal{Q}}) = \frac{\sum_{d_i \in \mathcal{D}} (D_{KL}(\pi_{\mathcal{P}}(i)||\pi_{\mathcal{Q}}(i)) + D_{KL}(\pi_{\mathcal{Q}}(i)||\pi_{\mathcal{P}}(i)))}{2|\mathcal{D}|}$$

Definition 5.3.2. **Graph Cosine Similarity**. We also define a metric of similarity between two

diffusion graphs $\Pi_{\mathcal{P}}$ and $\Pi_{\mathcal{Q}}$, as the average cosine similarity [81] between their diffusion vectors.

$$Cos(\Pi_{\mathcal{P}}, \Pi_{\mathcal{Q}}) = \frac{1}{|\mathcal{D}|} \sum_{d_i \in \mathcal{D}} \frac{\pi_{\mathcal{P}}(i) \cdot \pi_{\mathcal{Q}}(i)}{||\pi_{\mathcal{P}}(i)|| \cdot ||\pi_{\mathcal{Q}}(i)||}$$

A better model should infer a diffusion graph that is closer to the "ground truth" (the simulated diffusion network), that is, a lower KL-divergence and a higher Cosine similarity.



(a) Effectiveness (Divergence) (b) Effectiveness (Similarity)

Figure 5.1: Diffusion Evaluation on the Synthetic Dataset

In practice, we calculate the two metrics for the result of each method and each theme[4], and connect the KL-divergence scores in decreasing order (Figure 5.1(a)) and Cosine-similarity scores in increasing order (Figure 5.1(b)). The aggregated performance of the $1,000$ themes is reported in the 1^{st} and 2^{nd} columns of Table 5.2. **We can conclude that *TIDE* achieves the best performance, then *TIDE-*, then *NetInf*, and then *IndCas***.

Analysis on Content Evolution

In this experiment, we study how successfully TIDE and the baseline models track topic evolution. Since *NetInf* and *IndCas* are not able to handle topics, we compare our models *TIDE* and *TIDE-* with a simple mixture model stated in [95].

We repeat similar experiments as done in Section 5.3.2. We use two similar metrics (*i.e.*, the symmetrized KL-divergence and the Cosine similarity) to measure the closeness of the discovered

[4]To make the results from all methods comparable, vertices associated with topic snapshots are removed from the diffusion graphs inferred by *TIDE* and *TIDE-*.

word distributions of the topic snapshots to the "ground truth" (topic snapshots we construct in the synthetic dataset). The results are reported in Table 5.1.

Metric	KLD	CS
FM	0.4281	0.7033
TIDE-	0.3301*	0.8622*
TIDE	**0.2893***	**0.8774***

Table 5.1: Evolution Evaluation on the Synthetic Dataset

(KLD = Kullback-Leibler Divergance, CS = Cosine Similarity, FM = Feedback Model [95])

* means the improvement (over the above row) hypothesis is accepted at the significance level 0.001 based on dependent t-test.

As shown above, *TIDE* outperforms the other two methods with sufficient certainty, which proves our statement above: **the evolution and the diffusion of topics are compound processes; the success of one aspect will help the inference of the other**.

Proof of Combined Power

With this experiment, we can also prove that both social networks and text information play an important role the inference of topic diffusion.

Object	TDG		$\mathcal{H}$		$\mathcal{G}$	
Metric	GKLD	GCS	GKLD	GCS	GKLD	GCS
NetInf	0.0936	0.9359	1.2906	0.8685	0.9490	0.8022
IndCas	0.1601	0.9313	1.2971	0.8550	**0.7210**	**0.8975**
TIDE–	0.0628*	0.9691*	**0.9906**	**0.9494**	0.8757	0.8407
TIDE	**0.0524***	**0.9722***	1.0109	0.9378	0.8459	0.8547

Table 5.2: Diffusion Evaluation on the Synthetic Dataset

(TDG = True Diffusion Graph, GKLD = Graph Kullback-Leibler Divergence, GCS = Graph Cosine Similarity)

We measure the statistical significance of the improvement using the dependent t-test. * means that the improvement (over the row above) hypothesis is accepted at significance level 0.001.

First, we create a document network (denoted as $\mathcal{H}$), where the edge weight is proportional to the content similarities between documents. We compare each inferred diffusion graph with $\mathcal{H}$, and report the aggregated value of the two metrics in the 3^{rd} and 4^{th} columns of Table 5.2.

Second, we project each diffusion graph Π into a user network (denoted as $f(\Pi)$), compare $f(\Pi)$ with the general social network $\mathcal{G}$, and report the aggregated value of the two metrics in the last two columns of Table 5.2.

We can observe some phenomena that accord with our hypothesis in designing our model: *TIDE-* infers diffusion graphs only considering textual information without considering the social network structure, while *IndCas* infers the diffusion network purely based on the social influences. Indeed, the diffusion networks inferred by *TIDE-* are significantly biased towards the document similarity networks $\mathcal{H}$, and the diffusion networks inferred by *IndCas* are biased towards the social networks $\mathcal{G}$. **Neither of them infers diffusion networks that are closer to the ground truth than TIDE, which employs both text information and the social network**.

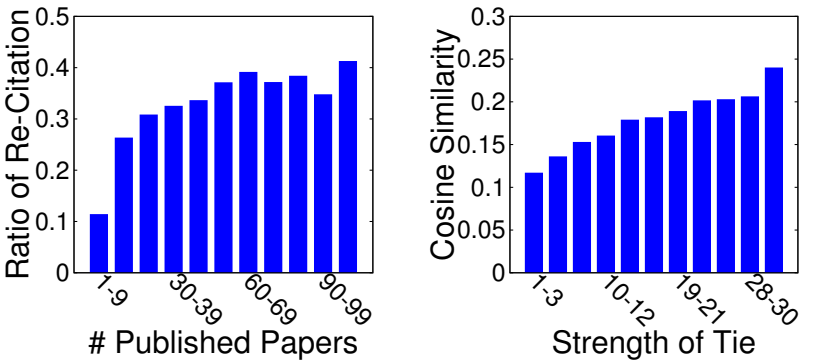
5.3.3 Experiments on Real Social Networks

We present the experiments on real world social communities in this section. Note that "ground truth" diffusion networks and topic snapshots are usually not available.

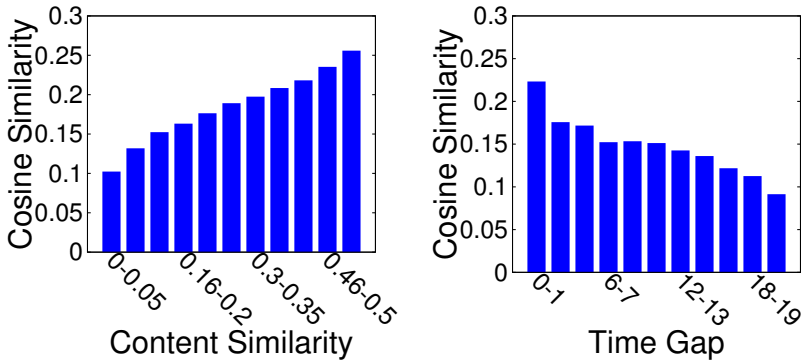
Verifying Motivating Observations

We start with the verification of the authenticity of the three motivating observations stated in Section 5.2.1. We expect that social influence, so that an author is more likely to adopt topics from the documents of her social connections. If this is the case, an author will pay consistent attention to papers published by authors she knows, or she has cited before. One intuitive way to verify this is through the behavior of 're-citation', *i.e.*, once the author cited one paper, it is likely that she will cite the paper of the same author again. We group authors by the number of publications, and plot the average ratio of re-citation in Figure 5.2(a). It shows that there are substantial re-citation behaviors, when an author publish more papers, the ratio of re-citation also grows. This verifies the existence of social influence in document-level information diffusion.

Instead of inferring the diffusion, a rough proxy of the influence of a cited paper d_r on a



(a) Existence of Ties (b) Diffusion and Ties



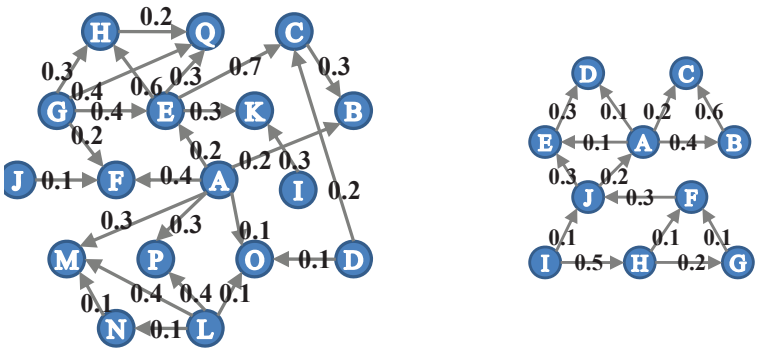
(c) Diffusion and Text (d) Diffusion and Time

Figure 5.2: Verifying Observations by DBLP-Citation Dataset

citing paper d_c can be measured by the author's behaviors after the citation. Generally, if the authors of d_c publish many papers related to d_r after they publish d_c, it is fair to believe d_r is quite influential to d_c. We partition the citations (each of which is recognized by a cited paper d_r and a citing paper d_c) into different groups according to the strength of the social connection between their authors. For each citation, we then compute the average document similarity between d_r and all papers published by d_c's authors after they had published d_c. The aggregated similarity is plotted in Figure 5.2(b). We repeat the same experiment, but partitions citations by degrees of the content similarity of d_r and d_c (Figure 5.2(c)), as well as the time gap (5.2(d)) between d_r and d_c. Figure 5.2(b)-5.2(d) prove our motivations that the (proxy) influence between two documents increases with the strength of social ties (Observation 5) and the content similarity, but decays over time (Observation 6).

Case Study

We select two themes for case study: one is about the research topic *'frequent pattern mining'* on the DBLP-Citation dataset, and the other is about the movie *'inception'* on the Twitter dataset.



(a) Theme 1 (DBLP-Citation) (b) Theme 2 (Twitter)

Figure 5.3: Case Study on Real Networks: Diffusion Graphs

ID	Publication	ID	Publication
A	J. Han, *etc*, SIGMOD'00.	B	A. Khan, *etc*, KDD'10.
C	X. Yan, *etc*, SIGMOD'04.	D	M. Zaki, *etc* KDD'03.
E	X. Yan, *etc*, KDD'03.	F	Y. Chi, *etc*, TKDE'05.
G	M. Zaki, KDD'02.	H	A. Bifet, *etc*, KDD'08.
I	X. Yan, *etc*, KDD'05.	J	U. Rükert, *etc*, SAC'04.
K	C. Chen, *etc*, CIKM'08.	L	J. Wang, *etc*, KDD'03.
M	J. Wang, *etc*, TKDE'05.	N	F. Pan, *etc*, KDD'03
O	A. Lee, *etc*, Infomation System'10.		
P	U. Yun, Knowledge-Based System'08		
Q	J. Balcázar, *etc*, Machine Learning'10.		

Table 5.3: Publications Shown in Figure 5.3(a)

Analysis on Information Diffusion. For theme 1, we apply the *TIDE* model on 344 papers published during the past ten years (2000 to 2010), which contain at least three primitive keywords in the title or abstract. A subgraph of the diffusion graph estimated by TIDE is shown in Figure 5.3(a), on a subset of 17 selected papers (listed in Table 5.3). The volume of each diffusion flow is marked on the edge. To quantitatively access the result, we compare the graph with three alternative "diffusion graphs." In the first graph, the weight of an edge $d_r \rightarrow d_c$ is set proportional

ID	Tweet (incomplete)
A	Inception had better special effects than Videodrome.
B	Inception's effects might take some Oscars.
C	I predict Inception's 12 Oscar nominations.
D	It has to be like a 3rd level Inception dream.
E	I wonder what level of recursive dreams.
F	Inception. What a brilliant, mind-twisting movie.
G	Watching inception. Long movie.
H	You'd be odd on twitter if you haven't seen Inception.
I	First time I have seen a movie in a theater in the last 6 months.
J	If you like intelligent movies and complex plots, go to see Inception.

Table 5.4: Tweets Shown in Figure 5.3(b)

to the total length of citation sentences where d_c mentions d_r. We then employ two experts to manually score the impact of each reference paper in a scale from one to five. The **Mean Absolute Error** [72], as the statistical metric of accuracy, based on each criteria, and the **Cohen's Kappa Coefficiency** [76], as the measure of inter-criteria agreement, are reported in Table 5.5.

MAE	SL	Exp1	Exp2
TIDE	0.1217	**0.1080**	0.1195

CKC	Exp1	Exp2
SL	0.5019	0.2095
Exp1	–	**0.6333**

Table 5.5: Accuracy Evaluation of Information Diffusion on DBLP-Citation Network

(MAE = Mean Absolute Error, CKC = Cohen's Kappa Coefficiency, SL = Sentence Length, RR = Replying Relation)

Theme 2 has been used as the running example mentioned above, and let us reveal more details. We apply the *TIDE* model on 361 tweets containing the keyword *'inception'*, and draw the diffusion graph on 10 selected tweets (listed in Table 5.4) in Figure 5.3(b). We repeat the same evaluation procedure as done for theme 1 (see Table 5.6), only except that the edge weight of the first criteria graph is decided by whether one tweet was replying the other.

In both cases, the opinions of the first expert gains the most agreement from others, and our result has the highest accuracy against the truth ground supplied by the first expert.

Analysis on Content Evolution.

We apply both *TIDE* and the feedback model [95] to extract the topic snapshots for two themes.

MAE	RR	Exp1	Exp2
TIDE	0.3632	**0.1301**	0.1351

CKC	Exp1	Exp2
RR	0.3583	0.3726
Exp1	–	**0.7500**

Table 5.6: Accuracy Evaluation of Information Diffusion on Twitter Network

(MAE = Mean Absolute Error, CKC = Cohen's Kappa Coefficiency, SL = Sentence Length, RR = Replying Relation)

Top words (with probabilities) of several selected topics are listed in Table 5.7-5.11. Note, to demonstrate more results, the word *'frequent'*, *'pattern'* and *'mining'* are eliminated from Table 5.7 and 5.8; and the word *'inception'* are eliminated from Table 5.10 and 5.11.

5.4 Conclusion

In this chapter, we propose *TIDE*, a novel probabilistic model for the joint inference of diffusion and evolution of topics in social communities. TIDE integrates the generation of text, the evolution of topics, and the social network structure in a unified model. Given the primitive form of any arbitrary topic, *TIDE* effectively tracks the topic snapshots that evolves along time and reveals the latent diffusion paths of the topic. Comprehensive experiment studies on both synthetic data and two real-world datasets show that TIDE outperforms existing approaches.

One important finding is that the discovery of topic diffusion and topic evolution benefits significantly from the joint inference process. Social influence still plays an important role in the diffusion of topics. Both text information and the general social network structure play an irreplaceable role to the inference process.

Primitive Topic		Year 2003		Year 2005		Year 2009	
frequent	0.20	itemset	0.05	itemset	0.04	itemset	0.03
pattern	0.40	GSM	0.03	tree	0.02	tree	0.02
mining	0.20	association	0.02	parallel	0.01	sequence	0.01
graph	0.05	apriori	0.02	graph	0.01	graph	0.01
tree	0.05	tree	0.01	sequence	0.01	slide	0.01
sequence	0.05	graph	0.01	traversal	0.01	gram	0.01
itemset	0.05	subgroup	0.01	optimize	0.01	window	0.01
		sequential	0.01	suffix	0.01	apriori	0.01

Table 5.7: Topic Snapshots by *TIDE* on Theme 1 (DBLP-Citation)

Year 2003		Year 2005		Year 2009	
efficient	0.02	close	0.01	sequential	0.02
close	0.01	itemset	0.01	itemset	0.01
association	0.01	match	0.01	tree	0.01
support	0.01	tree	0.01	graph	0.01
query	0.01	graph	0.01	database	0.01
temporal	0.01	sequential	0.01	efficient	0.01
graph	0.01	efficient	0.01	rule	0.01
rule	0.01	application	0.01	match	0.01

Table 5.8: Topic Snapshots by [95] on Theme 1

Table 5.9: Case Study on DBLP-Citation Networks: Topic Evolution

Primitive Topic		Jul 16-19		Jul 20-23		Jul 24-27	
inception	1.00	watch	0.05	dream	0.06	oscar	0.04
		night	0.05	mind	0.05	effect	0.04
		movie	0.05	level	0.03	dream	0.02
		special	0.03	walk	0.01	clever	0.01
		enjoy	0.01	recursive	0.01	briliant	0.01

Table 5.10: Topic Snapshots by *TIDE* on Theme 2 (Twitter)

Jul 16-19		Jul 20-23		Jul 24-27	
movie	0.06	type	0.05	oscar	0.03
night	0.06	eye	0.05	act	0.03
special	0.03	watch	0.05	dream	0.03
watch	0.03	night	0.05	strong	0.02
bad	0.03	dream	0.04	night	0.02

Table 5.11: Topic Snapshots by [95] on Theme 2

Table 5.12: Case Study on Twitter Networks: Topic Evolution

Chapter 6

Conclusion and Summary

The prevailing of Web 2.0 techniques has led to the boom of various online communities. Good examples are social communities such as Twitter, Facebook, Google+, and LinkedIn, which successfully facilitate the information creation, sharing, diffusion, and evolution among web users. As a result, a popular topic or event can spread much faster than in the Web 1.0 age. Indeed, when searching for a recent popular event (e.g., Hurricane Irene or Toyota Recall) on Twitter, all the results returned on the first page are created within the past five minutes.

In this thesis, I advance the data mining technique to create a system that detects, tracks, and analyzes the evolution and diffusion of popular events in a social community. Specially, in the first part of the dissertation, I introduce a mining algorithm for popular event detection, which can efficiently and effectively extract widely adopted and meaningful patterns of user behaviors; in the second part, I depict a novel and principled probabilistic model to track the popularity index of events in a time-variant social community that consists of both dynamic textual and structural information; in the third part of the dissertation, I address the problem of topic diffusions by studying the joint inference of topic diffusion and evolution in social communities, where contents and linkages in user-generated text information, together with social network structures, are used to facilitate the identification of topic adoption, the tracking of topic evolution, and the estimation of actual diffusion paths of any arbitrary topic.

References

[1] A. G. 0002, F. Bonchi, and L. V. S. Lakshmanan, "Learning influence probabilities in social networks," in *WSDM*, 2010, pp. 241–250.

[2] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social Networks*, vol. 25, no. 3, pp. 211–230, 2003.

[3] C. C. Aggarwal, Y. Xie, and P. S. Yu, "Gconnect: A connectivity index for massive disk-resident graphs," *PVLDB*, vol. 2, no. 1, pp. 862–873, 2009.

[4] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *SIGMOD Conference*, 1993, pp. 207–216.

[5] L. Araujo, J. A. Cuesta, and J. J. M. Guervós, "Genetic algorithm for burst detection and activity tracking in event streams," in *PPSN*, 2006, pp. 302–311.

[6] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, "Group formation in large social networks: membership, growth, and evolution," in *KDD*, 2006, pp. 44–54.

[7] L. Backstrom, D. P. Huttenlocher, J. M. Kleinberg, and X. Lan, "Group formation in large social networks: membership, growth, and evolution," in *KDD*, 2006, pp. 44–54.

[8] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *ICML*, 2006, pp. 113–120.

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," in *NIPS*, 2001, pp. 601–608.

[10] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," in *SIGMOD Conference*, 1997, pp. 265–276.

[11] J. J. Brown and P. H. Reingen, "Social ties and word-of-mouth referral behavior," *Journal of Consumer Research: An Interdisciplinary Quarterly*, vol. 14, no. 3, pp. 350–362, 1987.

[12] D. Cai, Q. Mei, J. Han, and C. Zhai, "Modeling hidden topics on document manifold," in *CIKM*, 2008, pp. 911–920.

[13] C. Chatfield, "The analysis of time series," in *Chapman and Hall*, 1984.

[14] C. Chen, C. X. Lin, X. Yan, and J. Han, "On effective presentation of graph patterns: a structural representative approach," in *CIKM*, 2008, pp. 299–308.

[15] L. Chen, Y. Hu, and W. Nejdl, "Deck: Detecting events from web click-through data," in *ICDM*, 2008, pp. 123–132.

[16] L. Chen and A. Roy, "Event detection from flickr data through wavelet-based spatial analysis," in *CIKM*, 2009, pp. 523–532.

[17] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *KDD*, 2010, pp. 1029–1038.

[18] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *KDD*, 2009, pp. 199–208.

[19] K. W. Church and W. A. Gale, "Poisson mixtures," vol. 1, no. 3, 1995, pp. 163–190.

[20] R. Crane and D. Sornette, "Robust dynamic classes revealed by measuring the response function of a social system," *the National Academy of Sciences*, vol. 105(41):15649-15653, Oct 2008.

[21] L. Dietz, S. Bickel, and T. Scheffer, "Unsupervised prediction of citation influences," in *ICML*, 2007, pp. 233–240.

[22] D. Easley and J. Kleinberg, "Networks, crowds, and markets: Reasoning about a highly connected world," *Cambridge University Press*, 2010.

[23] T. L. Fond and J. Neville, "Randomization tests for distinguishing social influence and homophily effects," in *WWW*, 2010, pp. 601–610.

[24] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu, "Parameter free bursty events detection in text streams," in *VLDB*, 2005, pp. 181–192.

[25] S. Gerrish and D. M. Blei, "A language-based approach to measuring scholarly impact," in *ICML*, 2010, pp. 375–382.

[26] M. Gomez-Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," in *KDD*, 2010, pp. 1019–1028.

[27] D. Gruhl, R. V. Guha, D. Liben-Nowell, and A. Tomkins, "Information diffusion through blogspace," in *WWW*, 2004, pp. 491–501.

[28] D. Gunopulos and G. Das, "Time series similarity measures and time series indexing," in *SIGMOD*, 2001, p. 624.

[29] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M. Hsu, "Freespan: frequent pattern-projected sequential pattern mining," in *KDD*, 2000, pp. 355–359.

[30] Q. He, K. Chang, and E.-P. Lim, "Analyzing feature trajectories for event detection," in *SIGIR*, 2007, pp. 207–214.

[31] Z. He, X. Xu, and S. Deng, "Mining top-k strongly correlated item pairs without minimum correlation threshold," *KES Journal*, vol. 10, no. 2, pp. 105–112, 2006.

[32] T. Hofmann, "Probabilistic latent semantic analysis," in *UAI*, 1999, pp. 289–296.

[33] A. T. Ihler, J. Hutchins, and P. Smyth, "Adaptive event detection with time-varying poisson processes," in *KDD*, 2006, pp. 207–216.

[34] L. Jiang, D. Yang, S. Tang, X. Ma, and D. Zhang, "Tight correlated item sets and their efficient discovery," in *APWeb/WAIM*, 2007, pp. 74–82.

[35] M. Karnstedt, D. Klan, C. Pölitz, K.-U. Sattler, and C. Franke, "Adaptive burst detection in a stream engine," in *SAC*, 2009, pp. 1511–1515.

[36] Y. Ke, J. Change, and J. X. Yu, "Efficient discovery of frequent correlated subgraph pairs," in *ICDM*, 2009, pp. 239–248.

[37] Y. Ke, J. Cheng, and W. Ng, "Mining quantitative correlated patterns using an information-theoretic approach," in *KDD*, 2006, pp. 227–236.

[38] Y. Ke, J. Cheng, and J. X. Yu, "Top-k correlative graph mining," in *SDM*, 2009, pp. 1038–1049.

[39] W.-Y. Kim, Y.-K. Lee, and J. Han, "Ccmine: Efficient mining of confidence-closed correlated patterns," in *PAKDD*, 2004, pp. 569–579.

[40] M. Kimura, K. Saito, and H. Motoda, "Minimizing the spread of contamination by blocking links in a network," in *AAAI*, 2008, pp. 1175–1180.

[41] M. Kimura, K. Saito, K. Ohara, and H. Motoda, "Learning to predict opinion share in social networks," in *AAAI*, 2010.

[42] D. Klan, M. Karnstedt, C. Pölitz, and K.-U. Sattler, "Towards burst detection for non-stationary stream data," in *LWA*, 2008, pp. 57–60.

[43] J. M. Kleinberg, "Bursty and hierarchical structure in streams," in *KDD*, 2002, pp. 91–101.

[44] D. E. Knuth, "The art of computer programming: Sorting and searching," in *Addison-Wesley*, 1968.

[45] S. Kullback and R. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[46] T. Lappas, B. Arai, M. Platakis, D. Kotsakos, and D. Gunopulos, "On burstiness-aware search for document sequences," in *KDD*, 2009, pp. 477–486.

[47] C. Lee, H. Kwak, H. Park, and S. B. Moon, "Finding influentials based on the temporal order of information adoption in twitter," in *WWW*, 2010, pp. 1137–1138.

[48] Y.-K. Lee, W.-Y. Kim, Y. D. Cai, and J. Han, "Comine: Efficient mining of correlated patterns," in *ICDM*, 2003, pp. 581–584.

[49] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," *TWEB*, vol. 1, no. 1, 2007.

[50] J. Leskovec, M. McGlohon, C. Faloutsos, N. S. Glance, and M. Hurst, "Patterns of cascading behavior in large blog graphs," in *SDM*, 2007.

[51] S. Li, "Markov random field modeling in image analysis," in *Springer-Verlag New York, Inc.*, 2001.

[52] W. Li and A. McCallum, "Pachinko allocation: Dag-structured mixture models of topic correlations," in *ICML*, ser. ACM International Conference Proceeding Series, W. W. Cohen and A. Moore, Eds., vol. 148. ACM, 2006, pp. 577–584.

[53] Z. Li, B. Wang, M. Li, and W.-Y. Ma, "A probabilistic model for retrospective news event detection," in *SIGIR*, 2005, pp. 106–113.

[54] D. Liben-Nowell and J. Kleinberg, "Tracing the flow of information on a global scale using internet chain-letter data," *the National Academy of Sciences*, vol. 105, no. 12, pp. 4633–4638, 2008.

[55] C. X. Lin, Q. Mei, Y. Jiang, J. Han, and S. Qi, "Inferring the diffusion and evolution of topics in social communities," in *SNAKDD*, 2011.

[56] C. X. Lin, B. Zhao, Q. Mei, and J. Han, "Pet: a statistical model for popular events tracking in social communities," in *KDD*, 2010, pp. 929–938.

[57] L. Liu, J. Tang, J. Han, M. Jiang, and S. Yang, "Mining topic-level influence in heterogeneous networks," in *CIKM*, 2010, pp. 199–208.

[58] B. R. M. M. H. MacRoberts, "Problems of citation analysis," *Scientometrics*, vol. 36, no. 3, pp. 435–444, 1996.

[59] G. McLachlan and T. Krishnan, "The em algorithm and extensions," *Wiley series in probability and statistics, Hoboken*, 2008.

[60] Q. Mei, D. Cai, D. Zhang, and C. Zhai, "Topic modeling with network regularization," in *WWW*, 2008, pp. 101–110.

[61] Q. Mei and C. Zhai, "Discovering evolutionary theme patterns from text: an exploration of temporal text mining," in *KDD*, 2005, pp. 198–207.

[62] S. Morris, "Contagion," in *Review of Economic Studies*, 2000, pp. 57–78.

[63] R. Nallapati and W. W. Cohen, "Link-plsa-lda: A new unsupervised model for topics and influence of blogs," in *ICWSM*, 2008.

[64] R. Nickalls, "A new approach to solving the cubic: Cardan's solution revealed," in *The Mathematical Gazette*, 1993, pp. 354–359.

[65] N. Parikh and N. Sundaresan, "Scalable and near real-time burst detection from ecommerce queries," in *KDD*, 2008, pp. 972–980.

[66] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "Prefixspan: Mining sequential patterns by prefix-projected growth," in *ICDE*, 2001, pp. 215–224.

[67] D. Preston, P. Protopapas, and C. E. Brodley, "Event discovery in time series," in *SDM*, 2009, pp. 61–72.

[68] F. Qiu and J. Cho, "Automatic identification of user interest for personalized search," in *WWW*, 2006, pp. 727–736.

[69] T. Rattenbury, N. Good, and M. Naaman, "Towards automatic extraction of event and place semantics from flickr tags," in *SIGIR*, 2007, pp. 103–110.

[70] H. Rue and L. Held, "Gaussian markov random fields: Theory and applications - theory and application," *Chapman and Hall/CRC*, 2006.

[71] K. Saito, M. Kimura, K. Ohara, and H. Motoda, "Selecting information diffusion models over social networks for behavioral analysis," in *ECML/PKDD (3)*, 2010, pp. 180–195.

[72] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *WWW*, 2001, pp. 285–295.

[73] U. Schiefele, "Interest learning and motivation," in *Educational Psychologist*, 1991, pp. 299 – 323.

[74] Z. Shen, P. Luo, Y. Xiong, J. Sun, and Y. Shen, "Topic modeling for sequences of temporal activities," in *ICDM*, 2009, pp. 980–985.

[75] A. Si, H. V. Leong, and R. W. H. Lau, "Check: a document plagiarism detection system," in *SAC*, 1997, pp. 70–77.

[76] N. Smeeton, "Early history of the kappa statistic," *Biometrics*, vol. 41, p. 795, 1985.

[77] H. W. Sorenson, "Parameter estimation: Principles and problems," *Marcel Dekker*, 1980.

[78] M. Steyvers, P. Smyth, M. Rosen-Zvi, and T. L. Griffiths, "Probabilistic author-topic models for information discovery," in *KDD*, 2004, pp. 306–315.

[79] Y. Sun, J. Han, J. Gao, and Y. Yu, "itopicmodel: Information network-integrated topic modeling," in *ICDM*, 2009, pp. 493–502.

[80] P.-N. Tan, V. Kumar, and J. Srivastava, "Selecting the right objective measure for association analysis," *Inf. Syst.*, vol. 29, no. 4, pp. 293–313, 2004.

[81] P. Tan, M. Steinbach, and V. Kumar, "Introduction to data mining," vol. ISBN:0321321367, 2005.

[82] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "miner: extraction and mining of academic social networks," in *KDD*, 2008, pp. 990–998.

[83] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *KDD*, 2009, pp. 807–816.

[84] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Miner: extraction and mining of academic social networks," in *KDD*, 2008, pp. 990–998.

[85] X. Wan and J. Yang, "Learning information diffusion process on the web," in *WWW*, 2007, pp. 1173–1174.

[86] C. Wang, W. W. 0009, J. Pei, Y. Zhu, and B. Shi, "Scalable mining of large disk-based graph databases," in *KDD*, 2004, pp. 316–325.

[87] C. Wang, D. M. Blei, and D. Heckerman, "Continuous time dynamic topic models," in *UAI*, D. A. McAllester and P. Myllymäki, Eds. AUAI Press, 2008, pp. 579–586.

[88] X. Wang, C. Zhai, X. Hu, and R. Sproat, "Mining correlated bursty topic patterns from coordinated text streams," in *KDD*, 2007, pp. 784–793.

[89] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "Twitterrank: finding topic-sensitive influential twitterers," in *WSDM*, 2010, pp. 261–270.

[90] J. Whittaker, "Graphical models in applied multivariate statistics," in *Wiley Publishing*, 2009.

[91] D. Wu, G. P. C. Fung, J. X. Yu, and Q. Pan, "Stock prediction: an event-driven approach based on bursty keywords," *Frontiers of Computer Science in China*, vol. 3, no. 2, pp. 145–157, 2009.

[92] T. Wu, Y. Chen, and J. Han, "Association mining in large databases: A re-examination of its measures," in *PKDD*, 2007, pp. 621–628.

[93] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *ICDM*, 2002, pp. 721–724.

[94] N. B. Younes, T. Hamrouni, and S. B. Yahia, "Bridging conjunctive and disjunctive search spaces for mining a new concise and exact representation of correlated patterns," in *Discovery Science*, 2010, pp. 189–204.

[95] C. Zhai and J. D. Lafferty, "Model-based feedback in the language modeling approach to information retrieval," in *CIKM*, 2001, pp. 403–410.

[96] Q. Zhao, T.-Y. Liu, S. S. Bhowmick, and W.-Y. Ma, "Event detection from evolution of click-through data," in *KDD*, 2006, pp. 484–493.

[97] Q. Zhao, P. Mitra, and B. Chen, "Temporal and information flow based event detection from social text streams," in *AAAI*, 2007, pp. 1501–1506.

[98] D. Zhou, X. Ji, H. Zha, and C. L. Giles, "Topic evolution and social interactions: how authors effect research," in *CIKM*, 2006, pp. 248–257.

[99] Z. Zhou, Z. Wu, C. Wang, and Y. F. 0004, "Efficiently mining both association and correlation rules," in *FSKD*, 2006, pp. 369–372.

[100] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, 2003, pp. 912–919.

[101] Y. Zhu and D. Shasha, "Efficient elastic burst detection in data streams," in *KDD*, 2003, pp. 336–345.